VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

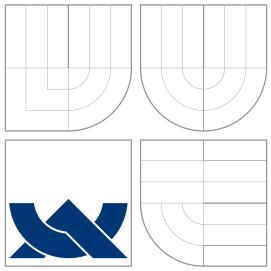
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DEMONSTRAČNÍ PROGRAM PRO UŽIVATELSKÁ ROZHRANÍ

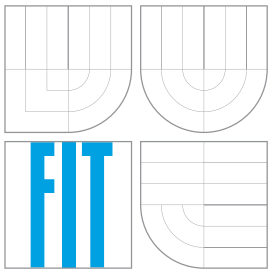
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MICHAL ADÁMEK
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DEMONSTRAČNÍ PROGRAM PRO UŽIVATELSKÁ ROZHRANÍ

USER INTERFACE DEMO PROGRAM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL ADÁMEK
AUTHOR

VEDOUCÍ PRÁCE — Doc. Dr. Ing. PAVEL ZEMČÍK
SUPERVISOR

BRNO 2010

Abstrakt

Tato bakalářská práce se zabývá analýzou, návrhem a implementací programu, který slouží k demonstrování různých druhů uživatelských rozhraní na PC. Aplikace umožňuje vytváření úkolů a provádění testů těchto úkolů. Během plnění úkolu jsou zaznamenávány veškeré akce, které uživatel provedl.

Abstract

This bachelor's thesis describes analysis, design and implementation of an application for demonstrational program of various user interfaces on PC. The application allows creation of tasks and tests of these tasks. During the performance of the task are recorded all the actions that the user has made.

Klíčová slova

uživatelské rozhraní, GUI, testování, demonstrační program, Java

Keywords

user interface, GUI, testing, demo program, Java

Citace

Michal Adámek: Demonstrační program pro uživatelská rozhraní, bakalářská práce, Brno, FIT VUT v Brně, 2010

Demonstrační program pro uživatelská rozhraní

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Adámek
12. května 2010

Poděkování

Rád bych poděkoval svému vedoucímu Doc. Dr. Ing. Pavlu Zemčíkovi za vedení a poskytnuté věcné připomínky, které mi pomohly při zpracování této bakalářské práce.

Obsah

Kapitola 1

Úvod

Počítače jsou v dnešní době nedílnou součástí našich životů. Setkáváme se s nimi nejen v profesní sféře, ale využíváme je i v domácnosti, jak ve formě klasického osobního počítače, tak i ve formě různých spotřebních zařízení, o kterých jen málokdy můžeme tušit, že obsahují počítač, který řídí jejich činnost.

Tento trend je a bude neustále aktuální, protože naše společnost se dostala to věku, kdy je žádaná co možná nejvyšší produktivita práce, a proto je lidská práce postupně nahrazovaná prací strojů, které jsou řízeny počítačem. Tyto stroje vykonávají požadovanou činnost autonomně, tj. samostatně. Navzdory této vlastnosti se některé stroje neobejdou bez lidského zásahu. A tak člověk musí s daným zařízením (počítačem) komunikovat, tj. sdělovat mu požadavky, nastavovat parametry atd. K tomu, aby byla práce počítače, tedy i celého zařízení, účinná a efektivní je nutné, aby byly požadavky jednoznačně určené a správně nastavené.

S počítači pracují lidé různého intelektu, vzdělání, věku, pohyblivosti a přesvědčení, proto může vzniknout problém v komunikaci mezi člověkem a počítačem, a tím pádem je pravděpodobné, že dojde ke snížení efektivity práce jak počítače, tak člověka, který s ním komunikuje. Proto je žádoucí, aby se možnosti komunikace s počítačem neustále vyvíjely a zlepšovaly.

Cílem této bakalářské práce je vytvořit demonstrační program, který bude demonstrovat použití různých druhů uživatelského rozhraní na osobním počítači při řešení zadaného úkolu, dále pak během řešení daného úkolu zaznamenávat uživatelské akce, které byly provedeny během plnění úkolu.

Důvodem, proč jsem si vybral toto téma, byl můj zájem o uživatelská rozhraní, tedy převážně tvorba uživatelských rozhraní pro různé aplikace a také mě zajímala problematika tvorby demonstračních programů.

Práce je rozdělena na pět kapitol. V následující kapitole jsou uvedeny základní pojmy z oblasti uživatelských rozhraních. Je uvedena stručná historie vývoje uživatelských rozhraní, využívané prostředky pro tvorbu uživatelských rozhraní, možné způsoby zaznamenávání uživatelských akcí a současný stav uživatelských rozhraní a demonstračních programů pro uživatelská rozhraní. Kapitola třetí se zabývá analýzou problému a návrhem požadované aplikace. Čtvrtá kapitola popisuje použité implementační prostředky - využitý programovací jazyk, vybrané vývojové prostředí a ostatní prostředky, které byly využité při tvorbě aplikace. Dále uvádí postup implementace jednotlivých částí aplikace a testování výsledné aplikace. Závěrečná kapitola obsahuje zhodnocení dosažených cílů, využitelnost programu a jeho možné další vylepšení a rozšíření.

Kapitola 2

Uživatelské rozhraní

Tato kapitola popisuje základní uvedení do problematiky uživatelských rozhraní. Jsou zde uvedeny jen základní teoretické předpoklady, které jsou relevantní pro tuto bakalářskou práci.

2.1 Pojem uživatelské rozhraní

Pojem uživatelské rozhraní definuje množinu operací, postupů, metod…, pomocí kterých lze pracovat s objektem, pro který je tato množina definována. Uživatelské rozhraní tedy nesouvisí pouze s počítači, obecně s informatikou, ale zasahuje do všech oblastí lidského pole působnosti. Každá věc, zařízení - osobní počítač, mobilní telefon, okno, vysavač atd., používá pro interakci s člověkem (uživatelem) jistý druh rozhraní. Toto rozhraní je pro každé zařízení jiné, ale základní požadavky jsou stále stejné - komunikace s uživatelem za účelem vykonání nějaké činnosti.

Komunikační kanály[1]

Komunikace mezi uživatelem a daným zařízením probíhá pomocí komunikačních kanálů, tyto kanály vycházejí ze základních lidských smyslů, tj. hmat, zrak, sluch, čich a chuť. Komunikační kanály lze rozdělit na dva druhy, které se liší směrem přenosu informace - od uživatele k zařízení nebo od zařízení k uživateli. Požadavkem je technická realizovatelnost, spolehlivost a cena.

Pro přenos informace od uživatele k zařízení lze využít:

Hmat - ovládání zařízení pomocí pohybu. Mezi nejobvyklejší prostředek patří klávesnice v nejrůznějších podobách. V současnosti nejpoužívanější způsob předávání informací do zařízení.

Obraz - využití gest, pohybů či mimiky k ovládání zařízení. V dnešní době jsou již některé dílčí aplikace využity k rozpoznávání objektů v obraze, ke strojovému čtení apod. V budoucnosti možný doplněk pro virtuální realitu.

Zvuk - ke komunikaci je využita řeč uživatele. Uživatel komunikuje pomocí slovních příkazů. Největší problém je porozumění lidské řeči na straně zařízení.

[1] Převzato z [26].

Pro přenos informace opačným směrem, tj. od zařízení k uživateli lze využít:

Hmat - komunikace formou pohybu např. pomocí vibrací. Perspektivní využití s virtuální realitou. V dnešní době je využíváno tohoto typu při komunikaci s nevidomými uživateli.

Obraz - považuje se za nejvýhodnější typ, pomocí kterého se dají přenášet informace od zařízení k uživateli. Dovoluje vysokou propustnost informací a náhodný přístup k těmto informacím, jelikož jsou obsaženy v obraze.

Zvuk - vhodný pro přenos menšího počtu informací pro doplnění obrazu. Nevýhodou je nižší propustnost a sériový přenos informace. Výhodou je, že může na sebe upozornit.

Čich, chuť - v současnosti nepoužitelné - technologie není na dostatečné úrovni, aby dokázala realizovat tento typ komunikace. V budoucnosti možný doplněk pro virtuální realitu.

Základní pravidla

Existuje celá řada pravidel, heuristik a pokynů, podle kterých je možné se řídit při tvorbě uživatelských rozhraní. Níže jsou uvedeny pravidla, které zformuloval Ben Shneiderman a jsou známá jako: „Eight Golden Rules to Interface Design“ viz [3]. Tyto pravidla však nelze brát doslovně a absolutně. Poskytují pouze základní představu o návrhu a je třeba s nimi nakládat podle konkrétní situace, kde může mít smysl tato pravidla nějakým způsobem modifikovat - zpřísnit, rozšířit, změnit apod.

Jak je výše uvedeno, jedná se o osm základních pravidel:

Konzistence - podobné sekvence akcí v obdobných situacích. Využití identické terminologie u rozvržení rozhraní, příkazů atd. Vytváření stereotypů - podobné věci se provádějí podobně.

Použití zkratek - omezení počtu uživatelských akcí a tím zrychlení práce, využitím zkratek, funkčních kláves, skrytých příkazů apod. Jedná se o další alternativní způsob ovládání, využívaný zejména zkušenějšími a častějšími uživateli.

Zpětná vazba - reakce systému na podnět od uživatele. Na každou uživatelskou akci, by měl systém adekvátně reagovat, aby uživatel věděl, zda akce proběhla úspěšně či nikoliv. Rozlišují se dva druhy zpětné vazby. Prvním druhem je silná zpětná vazba. Na tuto zpětnou vazbu musí uživatel explicitně reagovat - potvrdit její obdržení nebo vybrat z nabízených možností. Další druh je slabá zpětná vazba, na kterou nemusí uživatel explicitně reagovat, tj. nemusí potvrzovat její obdržení. Při výběru zpětné vazby je vždy nutné vhodně vybrat její typ tak, aby nebyl uživatel příliš zaneprázdněn její obsluhou.

Uzavřenost akcí - posloupnosti akcí by měli být uzavřeny v uzavřených celcích, které mají začátek, tělo a konec. Po dokončení celku následuje adekvátní zpětná vazba.

Řešení chyb - předcházení chybám uživatele tak, aby uživatel nemohl udělat chybu. Pokud k chybě dojde, je nutné uživatele o nastalé situaci informovat a také je vhodné sdělit příčiny chyby a možné způsoby vyřešení tohoto problému.

Návratnost akcí - návrat do předešlého stavu. Tímto je umožněno experimentování se systémem a prozkoumávání nových možností systému. Usnadňuje řešení chyb a také tolerantnost k uživatelskému chybování. Při volání akcí je možnost návratu nebo ukončení volání akce. Typicky je řešeno pomocí funkce znovu, zpět apod.

Předvídatelnost - odpovědný za systém je uživatel. Iniciátor akci - řídící člen je uživatel, ten zadává požadavky, které jsou zpracovávány systémem, který je jejich vykonavatel, ne naopak.

Šetření paměti - krátkodobá paměť uživatele má omezenou kapacitu, proto je vhodné možná co nejpřehlednější rozhraní, aby si uživatel nemusel pamatovat, kde se nachází jaký prvek nebo nemusel neustále číst popisky či nápovědy.

Požadavky

Na uživatelské rozhraní jsou kladeny určité požadavky, tyto požadavky víceméně vycházejí z výše uvedených pravidel a dají se shrnout do jednoho slova, a to je „použitelnost“. Norma ISO 9241-11[2] definuje použitelnost jako:

> „The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.“

Existuje celá řada specifických definic použitelnosti pro uživatelská rozhraní. Mezi známé patří „5Es“ viz [18] od Whitney Quesenbery. Těchto „5Es“ může být využito více způsoby k vybudování úspěšného rozhraní i systému, který toto rozhraní používá. Lze je použít při definování priorit, uživatelských požadavků na rozhraní, apod.

Jak je výše uvedeno, jedná se o pět „E“[3]:

Effective - efektivita - ukazuje, zda je rozhraní užitečné a zda pomáhá uživateli splnit jeho cíle přesně tak, jak si přeje.

Efficient - vhodnost - rychlost a přesnost, jak uživatelé dokončí požadovanou úlohu.

Engaging - nadchnutí - jak je rozhraní příjemné, uspokojivé a zajímavé. Mezi klíčové vlastnosti patří vizuální vzhled, styl a kvalita zpracování rozhraní.

Error tolerant - tolerantnost chyb - prevence výskytu chyb a případné zotavení se z chybového stavu.

Easy to learn - snadnost naučení - jak podporuje oba druhy učení - prvotní seznámení i hlubší učení. Rozhraní může být použito pouze jednou, jednou za čas nebo denně a je jedno, pokud rozhraní používá expert nebo laik, základní funkční rozhraní musí být zapamatovatelné nebo naučitelné a nové možnosti (klávesové zkratky apod.) rozhraní mohou být zjištěni časem.

Při práci s pěti „E“ je nutné uvažovat je jako jeden vyvážený celek, protože jsou navzájem závislé. Nelze se zaměřit pouze na jeden prvek a dát mu velký význam, protože by se mohla snadno ztratit vyváženost, čímž by vznikl negativní výsledek.

[2] Původní anglické znění, viz [1].

[3] Jelikož se jedná o definici v anglickém jazyce, tak je ponechán původní anglický název a za pomlčkou je uveden český překlad.

Každý uživatel má jinou představu o použitelnosti rozhraní. Někdo preferuje efektivitu, někdo jiný zase snadnost naučení, proto je nutné před samotným návrhem zjistit cílovou skupinu uživatelů a vytvořit jistý kompromis „5Es“. Následný vývoj by měl být zaměřený na co nejlepší výsledek, proto je vhodné využít vhodný vývojový model, vytvářet prototypy a konfrontovat cílového uživatele s dosavadním vývojem.

2.2 Uživatelské rozhraní na osobním počítači

Od této části se bakalářská práce zabývá problematikou uživatelského rozhraní na osobním počítači. Dále v textu jsou popsány tyto témata: stručná historie vývoje, prostředky pro tvorbu uživatelských rozhraní, možné způsoby zaznamenávání uživatelských akcí a současný stav uživatelských rozhraní a demonstračních programů pro uživatelská rozhraní.

Historie

Historie uživatelského rozhraní na osobním počítači je v mnohém shodná se samotnou historií počítačů (výpočetní techniky), s postupným vývojem počítačů se vyvíjelo i rozhraní, pomocí kterého komunikovaly s uživatelem.

Historii rozhraní lze rozdělit na čtyři období: prehistorie, dávkové zpracování, rozhraní příkazového řádku a grafické uživatelské rozhraní; historii počítačů lze rozdělit na víc období. Informace v této části jsou čerpány z [21, 20, 23, 25, 24].

Prehistorie

Do této kategorie patří první zařízení, ze kterých se vyvinuly současné počítače. Tyto zařízení pracovala na mechanickém principu. Mezi takovéto zařízení patří např. abakus, který vznikl přibližně před 5000 lety a od něj odvozené další mechanické počítadla.

Dalším pokrokem byl objev logaritmických tabulek, které roku 1614 objevil John Napier. Z logaritmických tabulek bylo následně vyvinuto logaritmické pravítko, které se využívalo až do 70. let 20. století, kdy bylo nahrazeno elektronickými kalkulátory.

V roce 1623 sestavil Wilhelm Schickard první mechanický kalkulátor. Toto zařízení bylo sestaveno z ozubených koleček, které pocházeli z hodinových strojků, proto se někdy nazývá jako „počítací hodiny“. Uměl základní matematické operace (sčítání a odčítání) s šesticifernými čísly.

Teprve kolem roku 1820 byl vytvořen první úspěšně masově produkovaný mechanický kalkulátor, který vytvořil Charles Xavier Thomas, a nazýval se „Thomasův Arithmometr“. Tento přístroj uměl sčítat, odčítat, násobit a dělit.

Okolo roku 1725 využil Basile Bouchon děrovaného papíru k řízení tkalcovského stavu. Tuto metodu zdokonalil Jean-Baptiste Falcon spojením jednotlivých papírových karet, čímž vytvořil robustnější formu pro technologii řízení tkalcovských stavů.

V roce 1801 Joseph-Marie Jacquard vytvořil stav, který byl řízen pouze pomocí děrných štítků. Tyto děrné štítky se daly vyměňovat bez zásahu do mechaniky stavu. Tento rok je považován za počátek éry programování.

Po vynálezu děrných štítků bylo postaveno mnoho dalších zařízení, které využívaly této technologie. Ale až v roce 1890 naplno tuto technologii využil Herman Hollerith při sčítání lidu v USA. Od tohoto okamžiku se technologie děrných štítků využívala ve vyspělých zemích až do 70. let 20. století.

O uživatelském rozhraní v této éře nelze mnoho říci. Přístroje byly ovládány přímo, pomocí nastavování hodnot. Malou změnu přinesla až technologie děrných štítků, kde byly pro podporu práce vyvíjeny speciální zařízení - děrovače, třídiče apod., které sloužily k zjednodušení a urychlení práce.

Dávkové zpracování

Tato éra trvala přibližně mezi lety 1945 a 1968. V tomto období byl výpočetní výkon vzácný a drahý, proto bylo uživatelské rozhraní pouze základní. Uživatelé se museli přizpůsobovat počítačům a ne počítače uživatelům.

Vstupní informace (program i data) se předávaly pomocí děrných štítků, děrných pásek apod. Výstup mohl být na tiskárnu, děrný štítek, později i na magnetickou pásku. Jediná možnost „realtime“ interakce s počítačem byla pomocí řídící konzole, kterou měl na starost operátor systému.

Typický postup práce byl následující: na používaná média zapsat program a data; zápis se prováděl pomocí speciálního zařízení. Programovací jazyky, které sloužily k popisu programu byly velmi striktní, protože se využívaly co možná nejjednodušší překladače a interprety. Následovalo zpracování těchto médií na počítači, to trvalo nějakou dobu, protože se muselo čekat až budou dokončeny předchozí požadavky a operátor musel také často přidávat pomocné data nebo pomocný program k tomu, aby mohla být požadovaná úloha splněna. Po provedení operací byl k dispozici výstup, tento výstup obsahoval buď výsledek nebo chybové hlášení s popisem chyby.

Uživatelské rozhraní bylo v této éře velmi omezené, jak je výše uvedeno, jednalo se pouze o základní rozhraní, které nebylo nijak přizpůsobované člověku. Práce s počítačem spočívala v práci s děrnými štítky a vyhodnocením dosažených výsledků na straně uživatele. Na straně operátora se jednalo o práci s řídící konzolí stroje a s přídavnými periferiemi.

Postupem let se začaly zavádět tzv. „load-and-go“ systémy. Tyto systémy využívaly tzv. „monitor program“, který byl trvale zavedený v počítači. Uživatelské programy mohly volat jeho funkce, čímž se usnadňovala jejich tvorba. Mezi další výhody toto systému patřilo lepší a časnější odchytávání chyb, lepší a užitečnější zpětná vazba. Zavedení těchto systému znamenalo první krok k vytváření operačních systému a tudíž i k tvorbě uživatelských rozhraní, které se postupně začali orientovat na člověka.

Rozhraní příkazového řádku - Command-line interface - CLI

Dalším vývojovým stupněm se stalo rozhraní příkazového řádku. Toto rozhraní jako první pracovalo na principu dotaz - odpověď, kde se jako forma pro předávání informací využívaly textové příkazy.

V počátcích se tato technologie realizovala pomocí spojení počítače s dálnopisem, který se využíval cca do roku 1975, kdy byl nahrazen video terminály.

Z rozhraní příkazového řádku vzniklo textové uživatelské rozhraní. Zásadní rozdíl mezi příkazovým a textovým rozhraním spočívá v tom, že příkazové rozhraní využívá k výstupu nový řádek, zatímco textové rozhraní má obrazovku rozdělenou na rast, kde do každého bodu rastru může být vložen libovolný znak z podporované množiny znaků.

S nástupem rozhraní příkazového řádku došlo k rapidnímu zrychlení práce s počítačem, uživatelé mohou měnit své požadavky „okamžitě“ a systém jim „okamžitě“ odpoví - práce s počítačem „realtime“. Ale i nadále přetrvává nutnost učit se nové postupy, jak s počítačem komunikovat - příkazy, jejich syntaxi a sémantiku. Tento problém z části řeší textové rozhraní, které jako první vytváří prostředí se snahou uživateli co nejvíc zlehčovat práci.

Obě výše zmíněná rozhraní se používají dodnes. Příkazový řádek je využíván především experty, pro které představuje zrychlení práce a je využíván jako základní rozhraní pro komunikaci s mnoha různými zařízeními.

Grafické uživatelské rozhraní - Graphical user interface - GUI

Počátky GUI začínají v 60. letech 20. století, kdy na Stanford Research Institute pod vedením Douga Engelbarta vznikl systém „On-Line System (NLS)“ jako součást projektu Augmentation of Human Intellect. Systém NLS byl navržen, aby usnadňoval vytváření digitální knihoven - ukládání a vyhledávání elektronických dokumentů pomocí hypertextu. Uživatelské rozhraní bylo tvořeno textem, hypertextovými odkazy a vektorovou grafikou (pouze horizontální a vertikální čáry); k ovládání se využívala klávesnice, malá pěti znaková klávesnice tzv. „chord keyboard“ (celkem 2^5 funkcí) a myš, která byla dalším výsledkem výzkumu.

V roce 1973 představila firma Xerox PARC počítač Alto. Jednalo se o první počítač plně využívající GUI, který byl inspirovaný systémem NLS. Počítač byl ovládán klávesnicí a třítlačítkovou myší. Tento stroj byl na svou dobu převratný, využíval principů, které se používají i v dnešní době - paradigma „WIMP“ (W - Window - okno, I - Icon - ikona, M - Menu - menu, P - Pointing device - polohovací zařízení), které bylo také výsledkem vývoje firmy Xerox PARC. Ačkoliv byl tento počítač sebelepší, byla jeho cena příliš vysoká, aby se uplatnil na komerčním trhu.

Roku 1983 představila firma Apple Computer osobní počítač s názvem Apple Lisa, jehož systém byl inspirován výsledkem práce Xerox PARC, protože někteří členové tehdejšího týmu PARCu přešli k Apple Comupter. Uživatelské rozhraní systému požívalo pracovní plochu s ikonami (Desktop), drag and drop, rozbalovací menu apod. Dále byl zaveden nový prvek - koš. K ovládání se používala klávesnice a myš, u které se poprvé objevila možnost dvojkliku. Navzdory tomu, že tento projekt lze považovat za předka dnešních grafických uživatelských rozhraní, nebyl příliš komerčně úspěšný, tak jako mnoho dalších projektů, které vznikaly v této době, z důvodu vysoké ceny.

Prvním komerčně úspěšným počítačem s GUI se stal Apple Macintosh uvedený v roce 1984, který vycházel z počítače Apple Lisa, avšak jeho cena byla o mnoho nižší.

Od uvedení počítače Apple Macintosh nastal obrovský boom rozvoje GUI. Začaly vznikat nové produkty například VisiOn, systém vyvinutý IBM, Microsoft Windows 1.0 - 1985 apod.

Od 90. let 20. století se GUI začalo rozšiřovat na další typy zařízení - mobilní telefony, přenosné počítače apod.

Novodobé grafické rozhraní stále vycházejí z původních principů objevených před více než třiceti lety, avšak jsou dále vyvíjeny a rozvíjeny nové přístupy, které se snaží usnadnit práci s počítačem a také poskytnou uživateli nový zážitek jak při práci, tak při zábavě s počítačem.

2.3 Prostředky pro tvorbu uživatelských rozhraní

Tato část práce popisuje základní prvky (i s pokyny pro tvorbu) využitelné k tvorbě uživatelských rozhraní a tvorbu grafických uživatelských rozhraní.

Interakční styl

Existuje mnoho cest, jak může uživatel komunikovat s počítačem a také jak může počítač komunikovat s uživatelem. Tyto cesty lze nazvat jako interakční styl či druh uživatelského rozhraní. Interakční styl poskytuje oba dva hlavní prvky: vzhled a chování, které určují cestu, pomocí které spolu komunikují.

Existuje více kategorií interakčních stylů, každá kategorie má své výhody i nevýhody a je vždy vhodná jen pro jistý okruh uživatelů. Proto je vždy nutné vybrat správný interakční styl nebo správnou kombinaci více stylů.

Dle [5, 7] lze zavést následující rozdělení s pokyny pro tvorbu:

Příkazová řádka - první interaktivní způsob komunikace více viz 2.2. Mnohostranné a flexibilní, vhodné pro zkušené uživatele, kterým dovoluje vytvářet makra, zkratky apod. Nevýhodou je nutné učení se a zapamatování příkazů, jejich kombinací a použití.

Při tvorbě je vhodné se řídit podle následujících pokynů:

Smysluplné názvy příkazů - využití charakteristických a typických názvů příkazů. Například příkaz „add“ - přidá prvek a ne, že jej odstraní.

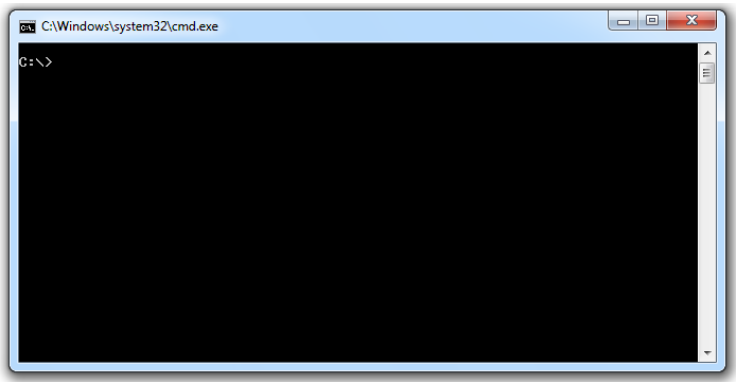
Délka příkazů - vytvářet co možná nejkratší, avšak informačně dostačující názvy příkazů.

Počet příkazů - omezení počtu příkazů a možných sekvencí příkazů k vyřešení požadovaného úkolu. Není nutné, aby k vyřešení jedné akce mohl uživatel použít „X“ různých příkazů.

Pravidla pro zkratky - vytváření odvoditelných zkratek příkazů.

Konzistentní syntaxe - všechny příkazy stejnou gramatickou strukturu.

Makra - vytváření maker umožňuje zrychlení práce. Makro je množina příkazů, která může být volána pomocí daného jména a její vykonání proběhne v jednom kroku. Je vhodné používat této možnosti pro často opakující se sekvence příkazů.



Obrázek 2.1: Windows 7 - Příkazová řádka.

Menu - jedná se o množinu voleb, z které si uživatel následně vybírá. Položky menu jsou zobrazeny jako text, ikony, případně kombinace, z kterých si uživatel pomocí ukazovacího zařízení (myš apod.), klávesnice vybere požadovanou položku. Menu jsou efektivní, snadno naučitelná a tudíž vhodná i pro laiky. Problém může nastat při vytváření rozsáhlých hierarchických menu, které zabírají velkou část obrazovky.

Při tvorbě je vhodné se řídit podle následujících pokynů:

Organizace - využívat sémantiku požadavků k organizování položek menu. „Soubor“, „Projekt“, „Nástroje“ apod. V submenu „Soubor“ je položka „Otevřít“, „Nový“ atd.

Smysluplné názvy položek - volit názvy položek menu tak, aby odpovídaly jejich funkčnosti.

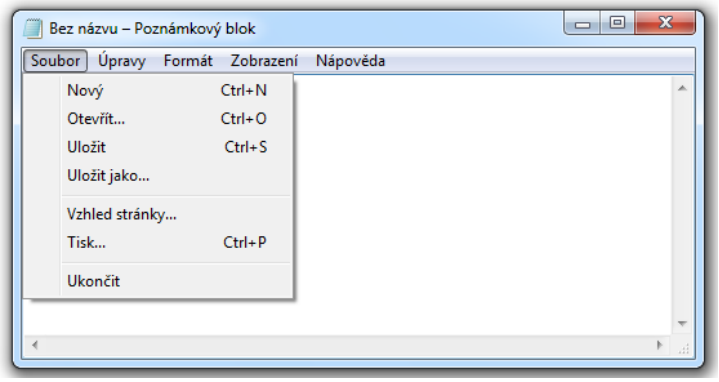
Slučování - slučovat položky menu do ucelenějších celků, ve který se snadněji vyhledává požadovaná položka.

Délka menu - dlouhé menu rozdělit do celků menších - zpřehlednění.

Délka názvu položky - volit vhodné názvy položek, které jsou co možná nejkratší.

Konzistence - využívat identickou gramatiku, rozložení a terminologii.

Kontextová nápověda - poskytnout kontextovou nápovědu k příkazům.



Obrázek 2.2: Windows 7 - Poznámkový blok - ukázka menu.

Vyplňování formuláře - vhodné pro sběr různých druhů informací od uživatele. Uživatel prochází formulářem a vyplňuje požadované vstupní hodnoty. Existuje více druhů vstupních polí: text, rozbalovací seznam, zaškrtávací tlačítka a případné kombinace.

Při tvorbě je vhodné se řídit podle následujících pokynů:

Popisky - použití smysluplných názvů položek.

Instrukce - poskytnout uživateli pochopitelné instrukce k vyplnění formuláře.

Slučování - vytváření logických sekvencí položek a jejich slučování do skupin.

Vzhled - využít vhodné rozložení položek a vhodné vizuální prvky.

Konzistence - využívat identickou gramatiku, rozložení a terminologii.

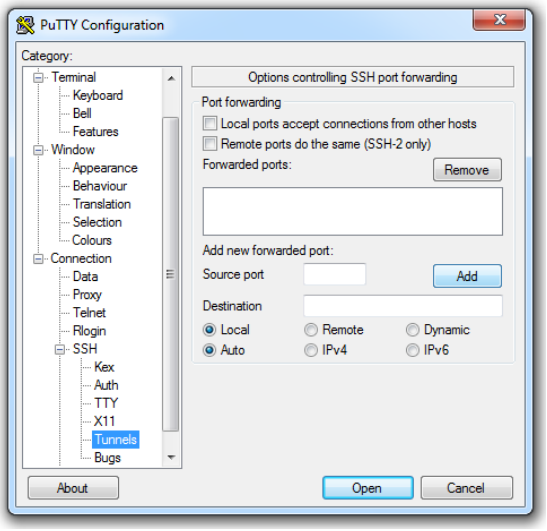
Defaultní hodnoty - poskytnout na výběr přednastavené hodnoty položek.

Oprava chyb - umožnit opravení špatně zadaných hodnot.

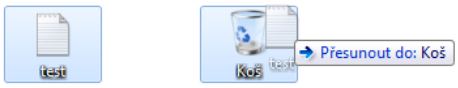
Chybové hlášení - v případě nezadaných či špatně zadaných hodnot poskytnout srozumitelné ohlášení chyb s místem výskytu chyby.

Povinné položky - explicitně označit položky, které je nutné zadat.

Nápověda - u popisků, u který by mohlo dojít k nepochopení od strany uživatele přidat doplňující text, který upřesní, co je od uživatele požadováno.



Obrázek 2.3: Windows 7 - Putty - ukázka formuláře.



Obrázek 2.4: Windows 7 - přímá manipulace.

Přímá manipulace - přímá manipulace s objekty rozhraní např. pomocí „drag and drop“ přesunutí souboru do koše. Využívá se v mnoha různých oblastech. Požadované operace s objektem se provádějí fyzickými akcemi - pohyby ukazovacími zařízeními, jako je přesunování, kliknutí, různá gesta. Operace jsou reprezentování textem, graficky (ikony), pomocí metafor nebo jejich kombinacemi. Tento způsob je snadně naučitelný a zapamatovatelný. Nevýhodou je, že je nutné mít odpovídající vstupní zařízení. Dále je nutné volit reprezentaci operací tak, aby nebyl jejich význam snadno zaměnitelný.

Při tvorbě je vhodné se řídit podle následujících pokynů:

Vizuální reprezentace - vytvoření vizuálních prvků, které reprezentují požadavky. Využití textu, ikon, metafor apod.

Vzhled - využít vhodné rozložení položek a vhodné vizuální prvky.

Zpětná vazba - poskytnout okamžitou zpětnou vazbu na vykonanou akci.

Akce - využívat rychlé, viditelné a zrušitelné akce.

Přirozený jazyk - komunikace s počítačem, jako s člověkem. Využití přirozeného jazyka - ovládání hlasem nebo psaní vět s požadavky, gest, mimiky, očního kontaktu apod. Jedná se o náročnou disciplínu, která se neustále vyvíjí. V současné době je omezeně využito převážně při komunikaci s postiženými uživateli.

Tvorba GUI

Téměř všechen moderní software, který používají uživatelé má grafické uživatelské rozhraní, které je tvořeno kombinací výše zmíněných interakčních stylů.

Při tvorbě GUI je vhodné se řídit podle pravidel, která nám určují, jak se má korektně vytvořit GUI pro zvolenou platformu či desktopové prostředí. Tyto pravidla jsou zapsána v tzv. „Interface Guidelines“ a existují téměř pro každou platformu či desktopové prostředí (například pro Windows 7 - [9], Mac OS X - [8], prostředí GNOME - [4]). V základu jsou si podobná, ale každá architektura má vlastní specifické rysy, které je vhodné dodržovat, pokud chceme vytvořit GUI, které splňuje pravidla pro danou technologii.

GUI je tvořeno pomocí mnoha základních jednotek (oken, dialogových boxů, tlačítek, list boxů apod.), které se souhrnně nazývají widgets - grafické komponenty. Tyto komponenty se mohou využívat samostatně nebo se mohou pomoci nich vytvářet nové komponenty, které jsou tvořeny vhodnou kombinací více základních.

Základní komponenty lze rozdělit na následující druhy:

Okna - základní prostředek při tvorbě GUI. Na okna jsou postupně rozmisťovány ostatní komponenty, které tvoří rozhraní.

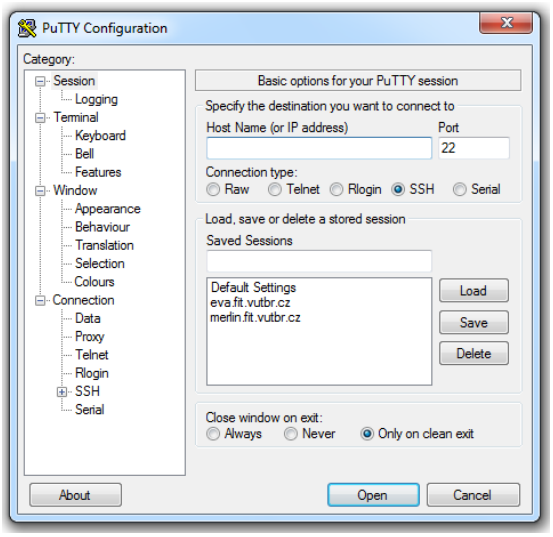
- **Primární okno** - hlavní okno aplikace. Bývá často jedno a jsou z něj vyvolávány závislá sekundární okna.
- **Sekundární okno** - slouží k rozšíření možností hlavního okna, poskytuje rozšířenou funkcionalitu a podporuje uživatele. Lze rozlišit dva hlavní druhy oken: závislá a nezávislá. Závislé sekundární okno je zobrazeno po příkazu z hlavní okna, které lze nazvat rodičovské. Pokud je rodičovské okno zavřeno, tak jsou zavřené také všechny okna na něm závislé (platí i pro minimalizaci). Nezávislé sekundární okno existuje samo o sobě a není závislé na primárním okně.

  Další dělení je na modální a nemodální. Modální sekundární okno nepovolí práci s jinými okny, dokud není samo zpracováno - obdoba silné zpětné vazby (např. chybové hlášení). Nemodální - obdoba slabé zpětné vazby, povoluje přechod na jiné okno.

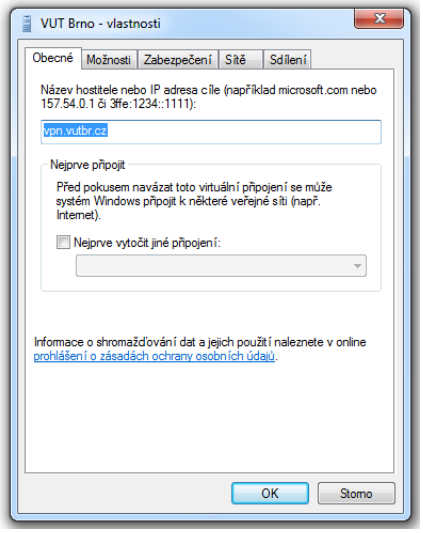
  - **Okna pro zprávy** - slouží k poskytování zpráv uživateli. Existuje více druhů zpráv: informační, upozorňovací, chybové a zpráva tázací.
  - **Dialogová okna** - jsou vyvolána na základě žádosti uživatele. Poskytují mu rozšiřující informace nebo jsou použity k celkovému vykonání úkolu. Lze je využít v případě, že je nutné zachovat posloupnost kroků - využití wizardu - průvodce.
- **Záložky** - rozděleni okna do více nezávislých celků pomocí záložkových karet.

Ovládací prvky - komponenty, které slouží k ovládání aplikace.

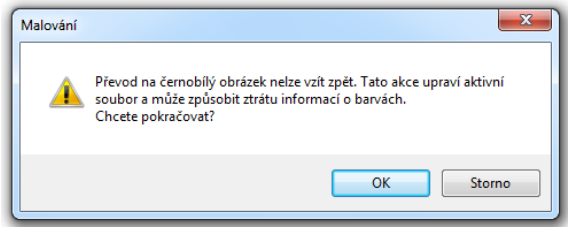
- **Menu** - existuje mnoho různých variant menu, ale všechny by měli do jisté míry splňovat požadavky uvedené v 2.3.
- **Panel nástrojů** - slouží k usnadnění a zrychlení práce, kdy jsou vybrané příkazy z menu reprezentovány pomocí obrázků - ikon. Panel nástrojů může být libovolně zarovnaný či se může jednat o samostatné okno.
- **Řídící tlačítka** - prvek pomocí, kterého je přímo vyvolána akce, například „Zpět“, „Tisk“, apod.



Obrázek 2.5: Windows 7 - Putty - hlavní okno aplikace.



Obrázek 2.6: Windows 7 - Nastavení síťového adaptéru - záložky.



Obrázek 2.7: Windows 7 - Malování - zpráva.

Vstupní prvky - komponenty, které slouží k zadávání informací. Uživatel pomocí nich předává nutné informace aplikaci.

Textové pole - vstup i výstup textové informace. Jelikož uživatel může vložit libovolné textové informace, je nutné předem zvážit, zda není vhodné využít nějakou jinou komponentu, která je pro daný případ vhodnější. Textové pole může být vybaveno automatickým doplňováním, validací vstupního textu apod.

Přepínače a volby - výběrová a zaškrtávací tlačítka. Reprezentují přesný a rychlý výběr z předem dané množiny prvků, která je neměnná.

Radio button - slouží k výběru jedné možnosti z více nabízených.

Check button - umožňuje žádné, jedné nebo i více možností současně.

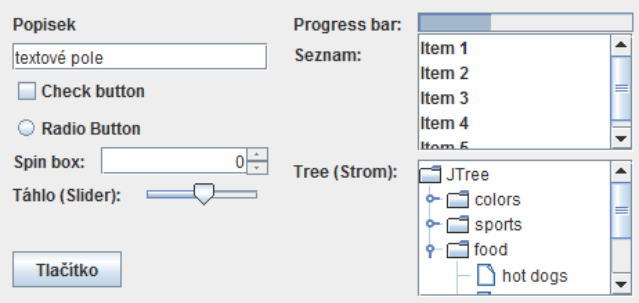
Seznamy - rozbalovací množiny možností a následný výběr. Výběr může být omezen na jeden či více prvků. Existuje mnoho různých variant seznamu: rozbalovací

Obrázek 2.8: Windows 7 - OpenOffice.org Writer - panel nástrojů.

seznam (Drop Down List Box), editovatelný rozbalovací seznam (Combo Box), strom (Tree) apod.

Táhla a Spin boxy - Spin box slouží k zadávání hodnot, které je možné pomocí integrovaných tlačítek inkrementovat či dekrementovat, vhodné je využití pro číselné hodnoty. Táhlo (Slider) - nastavování hodnot, která jsou jakoby „spojité“ - rychlost, hlasitost apod.

Ostatní prvky - jedná se o vizuální komponenty - popisky, rámečky, kontextová nápověda, oddělovače, ukazatel průběhu (Progress bar), ukazatele myši apod.



Obrázek 2.9: Windows 7 - Ukázka několika různých prvků.

Z výše uvedených základních prvků lze vytvářet složitější a sofistikovanější komponenty, které jsou následně vhodně využívány, čímž nedochází k opětovnému kombinování základních komponent - využití znovupoužitelnosti.

2.4 Zaznamenávání uživatelských akcí

Uživatelská akce je jakákoliv práce uživatele s rozhraním. Jednoduše je možné je rozdělit na akce myši, klávesnice a na akce provedené s aplikací. K zaznamenávání lze využít další osobu, která bude sledovat a vhodně zaznamenávat veškeré akce, které uživatel provedl. Získaná data mohou být nepřesná a nemusejí obsahovat záznamy o všech provedených akcí, proto je vhodnější využít automatického zaznamenávání akcí, které je přesnější a dovoluje zaznamenat veškeré akce.

Získané hodnoty lze následně vhodně vizualizovat, například pomocí tabulek, grafů, nebo je dále zpracovat pomocí vhodných statistických funkcí (modus, medián, průměr…). [22, 5]

2.5 Současný stav

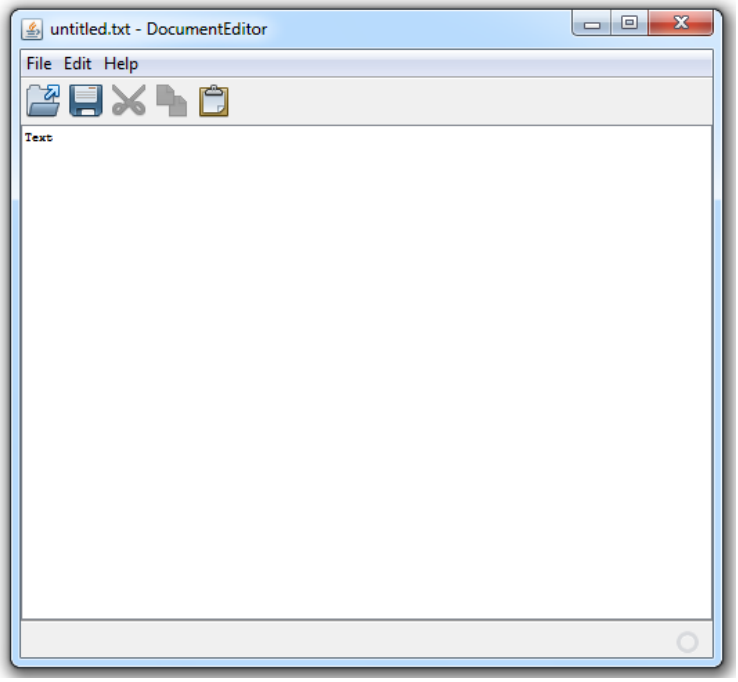
V současné době je nejvíce využívaným typem uživatelského rozhraní grafické uživatelské rozhraní. Tento druh rozhraní je nejjednodušší k naučení a následnému používání. Aplikace s GUI jsou použity téměř ve všech typech aplikací - operační systém, textový editor, systém pro řízení letového provozu apod. Spolu s GUI je také využíváno rozhraní příkazového řádku, které urychluje práci v prostředí s GUI. Dále existují i specifické aplikace, které využívají pouze příkazového řádku případně textového rozhraní pro komunikaci s uživatelem.

Čistě demonstračních programů, tedy programů, které prezentují možnosti, jak vytvářet či používat jisté uživatelské rozhraní existuje mnoho. Jedná se o ukázkové aplikace, pomocí

kterých se prezentují možnosti desktopového prostředí, softwarové platformy, programovacího jazyku či různého softwarového produktu. Tyto aplikace jsou více či méně propracované a snaží se uživateli ukázat, co daný produkt nabízí. Tyto aplikace lze rozdělit na dvě skupiny.

První skupinu tvoří tzv. zkušební aplikace, které mají omezenou funkcionalitu nebo jsou omezeny časově a slouží k prezentaci produktu. Uživatel si vyzkouší práci s touto aplikací a v kladném případě si pořídí plnohodnotnou a nijak funkčně neomezenou verzi. Aplikace, které mají omezenou funkcionalitu se často nazývají „demo“. Typickým představitelem aplikací této skupiny jsou demoverze her nebo aplikace se širokou paletou funkcí, kde v demoverzi nejsou některé funkce přístupné. Aplikace s časovým omezením se často nazývají „trial“. Omezení se může vztahovat na použití celé aplikace nebo jen na použití nějaké funkce např. pro uložení, tisk apod. Do této skupiny aplikací patří mnoho zkušebních verzí softwarových produktů - např. ESET NOD32 Antivirus 4 (více viz [2]).

Druhou skupinu tvoří aplikace, které prezentují možnosti jistého desktopového prostředí, softwarové platformy, programovacího jazyku - nejčastěji se jedná o aplikace, které jsou zdarma a jsou k dispozici i se zdrojovými kódy. Protože jsou tyto aplikace zdarma k dispozici i se zdrojovými kódy, tak často bývají využity ke studiu daného prostředí. [5, 11]



Obrázek 2.10: Ukázková aplikace s GUI v Javě. Program je k dispozici v rámci vývojového prostředí NetBeans.

Kapitola 3

Analýza a návrh řešení

Následující kapitola se zabývá analýzou problému a návrhem požadované aplikace.

3.1 Motivace

Nejvíce využívaným druhem uživatelského rozhraní je a v nejbližší době bude grafické uživatelské rozhraní s případným rozšířením o příkazový řádek, který bude usnadňovat a především zrychlovat práci zkušenějším uživatelům. Existující demonstrační programy jsou dle mého názoru dostačující k záměrům, ke kterým byly navrženy - tj. zkušební aplikace a aplikace, které prezentují možnosti jistého produktu a jsou zdarma i se zdrojovými kódy.

Cílem je vytvoření demonstrační aplikace, která bude demonstrovat použití různých druhů uživatelského rozhraní na osobním počítači při řešení zadaného úkolu, dále pak během řešení daného úkolu zaznamenávat uživatelské akce, které byly provedeny během plnění úkolu.

Motivací pro tvorbu této aplikace je můj zájem o uživatelská rozhraní, tedy převážně tvorba uživatelských rozhraní pro různé druhy aplikací a také mě zajímá problematika tvorby demonstračních programů, u kterých mě převážně zajímá, jakým způsobem bude uživatel s daným programem pracovat. Jak bude využívat jeho možnosti a jak bude schopný se s programem ztotožnit, aby využil jeho úplný potenciál.

Pro demonstrační aplikaci je nejprve nutné zvolit vhodný demonstrační příklad, kterým jsem po dohodě s vedoucím práce zvolil rozmísťování nábytku. Jedná se o umísťování různých druhů předmětů (židle, stůl, skříň…) do vymezeného prostoru. Dále je nutné zajistit možnost vytváření i plnění úkolu. Při plnění úkolu má uživatel za úkol volitelným způsobem umístit zvolený předmět na zadané souřadnice s nastaveným úhlem rotace daného předmětu. Při vytváření úkolu uživatel rozmístí předměty do vymezeného prostoru, provede popis úkolu a uloží jej. Pohled na předměty je realizovaný „shora“, tj. aplikace pracuje v 2D režimu.

3.2 Požadavky na aplikaci

Požadavky kladené na aplikaci lze tedy shrnout do níže uvedených bodů:

- Vhodný demonstrační příklad s možností vytváření testovacích úkolů.
- Různé typy uživatelského rozhraní pro ovládání demonstračního příkladu.

- Uživatelsky přívětivý vzhled aplikace.
- Zaznamenávání uživatelských akcí povedených při plnění úkolu a následná vizualizace výsledků.

Ovládání aplikace

Při výběru možných způsobů, jak ovládat aplikaci, jsem vycházel z informací uvedených v kapitole 2.3. Pro ovládání jsem tedy vybral následující způsoby:

- Ovládání pomocí myši - přímá manipulace s předměty pomocí myši.
- Ovládání pomocí klávesnice - celkové ovládání aplikace pomocí klávesnice a využívání klávesových zkratek.
- Příkazový řádek - využití příkazů pro práci s jednotlivými předměty.
- Dialogové okno - pro přidání nového předmětu.
- Panel dostupných předmětů.

3.3 Struktura aplikace

Pro návrh i implementaci je vhodné rozdělit aplikaci na dvě části - grafické uživatelské rozhraní a aplikační logiku. Část grafického uživatelského rozhraní bude poskytovat interakci s uživatelem, jelikož je nutné zabezpečit zaznamenávání akcí během plnění úkolu, které uživatel provede jak s předmětem, tak i s rozhraním, bude tato část dále zajišťovat zaznamenávání akcí provedených s rozhraním. Dále bude poskytovat grafickou reprezentaci prvků aplikační logiky. Mezi prvky části grafického uživatelského rozhraní patří tedy okna, grafická reprezentace předmětů, ovládací a podpůrné prvky pro práci s aplikací a prostředky pro zaznamenávání akcí provedených s uživatelským rozhraním.

Aplikační logika bude poskytovat funkcionalitu, která vychází z demonstračního příkladu a dále bude zajišťovat zaznamenávání akcí provedených s jednotlivými předměty.

3.4 Návrh grafického uživatelského rozhraní

Aplikace bude mít pouze grafické uživatelské rozhraní. Rozhraní pro klasický příkazový řádek nebude ani navrhováno, protože by se možnosti ovládání aplikace omezily pouze na zmíněný příkazový řádek. Aplikace bude tedy klasickou desktopovou aplikací s jedním hlavním oknem a více sekundárními okny.

Návrh hlavního okna

Hlavní okno a jím vyvolané sekundární okna budou jedinou možností interakce uživatele s aplikací. Obrázek 3.1 ilustruje prvotní návrh hlavního okna aplikace. Od tohoto návrhu se bude odvíjet vzhled hlavního okna výsledné aplikace. Výše zmíněná ilustrace rozděluje hlavní okno na několik částí:

Menu - hlavní a jediné menu celé aplikace. Obsahuje položky pro práci s aplikací, nastavování vzhledu aplikace, nápovědu a položky pro práci s úkolem.

Panel pro zobrazení úkolu a jeho řídících prvků - panel, ve kterém je zobrazen popis úkolu a ovládací prvky pro řízení testu úkolu.

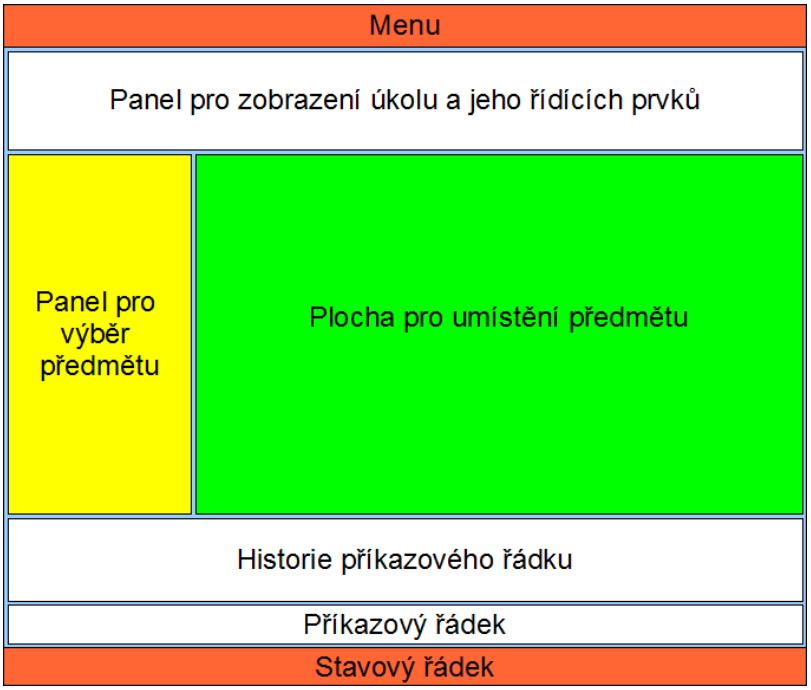
Panel pro výběr předmětu - seznam dostupných předmětů, které je možné využít při tvorbě úkolu a při zpracovávání zadaného úkolu.

Plocha pro umístění předmětu - plocha, na kterou jsou umísťovány vybrané předměty.

Historie příkazového řádku - slouží pro zobrazení historie akci prováděných s předměty a dále zobrazuje popis příkazu a nápovědu k příkazu.

Příkazový řádek - příkazový řádek využívaný aplikací, pomocí kterého jsou zadávány příkazy.

Stavový řádek - stavový řádek aplikace, který slouží k zobrazování stavových informací při práci s aplikací.



Obrázek 3.1: Prvotní návrh hlavního okna aplikace.

Sekundární okna

Aplikace bude obsahovat několik sekundárních oken - okna pro zprávy, dialogová okno pro nahrávání a ukládání souborů, okna pro rozšíření možností aplikace - okno pro nápovědu, okno pro přidání nového předmětu apod.

Vzhled aplikace

Aby byla aplikace uživatelsky přívětivá, bude mít uživatel možnost nastavit vzhled oken a hlavních barev využitých při práci s aplikací.

Zaznamenávání akcí

Existuje více druhů akcí, které může uživatel během plnění úkolu vyvolat. Lze je rozdělit na následující:

- Akce myši - pohyb, stisk tlačítka, držení tlačítka, uvolnění tlačítka, vstup, výstup, táhnutí, otočení kolečka.
- Akce kláves - akce kláves na klávesnici - stisknutí, uvolnění.
- Akce příkazového řádku - zda byl použitý příkaz validní, zda byl proveden.
- Ostatní akce - ostatní akce, které může uživatel během plnění úkolu provést. Otevření a zavření okna, nastavení hodnot apod.

U každého druhu akce se zaznamenávají různé informace:

- Akce myši - čas, prvek, typ operace, souřadnice x, souřadnice y.
- Akce kláves - čas, znak, modifikátory, název prvku.
- Akce příkazového řádku - čas, příkaz, stav.
- Ostatní akce - čas, vlastnost, nová hodnota, stará hodnota.

Z toho důvodu je vhodné mít pro každý druh záznamu vlastní třídu, která implementuje veškerou funkcionalitu nutnou k práci s daným druhem záznamu. Jednotlivé záznamy se ukládají do struktury s mapovací funkcí, kde jako klíč slouží automaticky generované číslo, čímž je zajištěna správná posloupnost událostí v pořadí, ve kterém vyvolaly akci.

Dále je potřeba prvek, který bude generovat klíč, pracovat se strukturou a bude schopen přidávat nové záznamy na základě požadavku na přidání - „listener“.

Získané záznamy budou následně vhodně vizualizovány pomocí tabulkového výpisu, který bude umožňovat filtrování a vyhledávání podle zvolených parametrů.

Grafická reprezentace prvků aplikační logiky

Z návrhu aplikační logiky budou reprezentovány grafickou podobou dva prvky. Prvním je plocha, na kterou se budou umísťovat jednotlivé předměty a dále bude obsahovat prvky, které zajistí korektní zobrazení a práci s touto plochou. Druhým je předmět, jeho grafická reprezentace se bude umísťovat na grafickou reprezentaci plochy.

3.5 Návrh aplikační logiky

Základním prvkem bude plocha, na kterou se budou umisťovat jednotlivé předměty. Plocha je omezený prostor, do kterého je možné umístit předmět. Jeden předmět bude reprezentovat jeden kus nábytku, který bude umístěný na daných souřadnicích a bude mít nastavenou rotaci. Každý předmět bude mít unikátní název, aby bylo možné jej snadno rozlišit a dále bude každý předmět mít vlastní seznam akcí, které s předmětem uživatel provedl.

Jelikož aplikace bude určená k vytváření a provádění testů - úkolů, tak je nutné vhodně navrhnout jejich strukturu. Úkol bude obsahovat název, popis, autora, datum uložení a seznam požadovaných typů předmětů, které je nutné umístit na plochu se zadanými parametry, aby byl úkol úspěšně splněn.

3.6 Datové formáty

Pro načítání a ukládání veškerých konfiguračních souborů a souborů, které popisují úkol a výsledky testu jsem vybral datový formát XML, který se používá jako standardní formát pro výměnu dat. Popis formátu viz [19].

Pro uložení obrázků jednotlivých předmětů jsem vybral formát PNG, který používá bezeztrátovou kompresi rastrové grafiky. Popis formátu viz [6].

Nápověda k aplikaci bude ve formátu XHTML. Popis formátu viz [17].

Kapitola 4

Implementace

Tato kapitola popisuje použité implementační prostředky - využitý programovací jazyk Java, vývojové prostředí a ostatní nástroje využité při tvorbě aplikace. Dále je uveden postup implementace jednotlivých částí a testování výsledné aplikace.

4.1 Implementační prostředky

Následující text popisuje implementační prostředky, které byly využity při tvorbě aplikace. Programovací jazyk Java jsem si vybral z níže popsaných výhod a dále proto, že jsem si chtěl zdokonalit mé dosavadní znalosti tohoto programovacího jazyka.

Programovací jazyk Java

Java byla představena roku 1995 firmou Sun Microsystems. V dnešní době patří do skupiny nejvíce využívaných programovacích jazyků[1]. Jedná se o jazyk, který je určený pro vytváření různých druhů aplikací. Lze vytvářet klasické desktopové aplikace (konzolové a v dnešní době „klasické“ aplikace s GUI), aplikace běžící na webovém serveru tzv. servlety a aplikace běžící v rámci internetového prohlížeče klientské stanice - aplety.

Mezi výhody jazyka patří:

Objektová orientovanost - využití objektově orientovaného přístupu k tvorbě aplikací.

Platformní nezávislost a přenositelnost - snadné přenášení mezi různými platformami, žádné problémy s druhy a verzemi operačního systému, knihoven a ostatních nutných náležitostí, které jsou nutné k běhu aplikace - snadná distribuce v rámci síťového prostředí.

Interpretovanost - souvisí s přenositelností. Aplikace není přeložena do strojového kódu, ale do mezikódu, tzv. byte-code (česky „bajtkódu“), který je interpretován pomocí virtuálního stroje Javy, tzv. Java Virtual Machine (JVM).

Robustnost - využívá silnou typovou kontrolu a automatickou správu paměti pomocí tzv. Garbage collectoru, který automaticky uvolňuje již nepotřebnou paměť, která byla programem alokována.

[1] dle Programming Language Popularity - http://www.langpop.com

Bezpečnost - autentifikace, autorizace, kryptografie, řízení přístupu k různým zdrojům, prostředkům, objektům apod.

Jako nevýhodu lze zmínit vyšší spotřebu paměti, protože je k běhu aplikace nutné celé běhové prostředí.

Jednou z důležitých části, kterou využívá Java pro svůj běh je Java Virtual Machine, což je množina počítačových programů, které zajišťují vazbu na hardware, interpretaci a optimalizaci byte-codu. Z důvodu zrychlení běhu aplikace začal být interpret nahrazován tzv. JIT (Just in time) kompilátorem, který během zavádění požadovaného programu přeloží byte-code do strojového jazyka daného zařízení. Takto přeložený byte-code následně běží stejnou rychlostí, jako jakákoliv jiná shodná aplikace, která byla zkompilována v jiném programovacím jazyku např. C++.

Mezi další důležitou část patří základní knihovna tříd - Java Core API[2]. Java Core API obsahuje mnoho knihovních tříd, které jsou považovány za standardní, proto je nutné, aby se byly k dispozici v každém prostředí, ve kterém je Java používána. Kromě Java Core API existuje celá řada dalších knihoven, které je možné při tvorbě aplikací využívat, ale tyto knihovny jsou již volitelné, takže se musí zajistit případná distribuce aplikace spolu s požadovanými knihovnami.

Java Virtual Machine a Java Core API jsou součástí nástrojů, které tvoří Java platformu. Java platforma je souhrnný název pro všechny nástroje nutné k tomu, aby bylo možné vytvářet a spouštět aplikace napsané v programovacím jazyce Java. Aplikace v jazyce Java je tedy přenositelná na každé zařízení, na kterém je připravena Java platforma. Tato platforma se dělí na dílčí platformy, které jsou specializovány na tvorbu různých druhů aplikací např[3].:

Java ME (Micro Edition) - platforma pro pro zařízení s omezeným výkonem - mobilní telefon, MDA, PDA apod.

Java SE (Standard Edition) - platforma pro aplikace provozované na klasických stolních počítačích.

Java EE (Enterprise Edition) - platforma pro podnikové aplikace a rozsáhlé informační systémy.

Java FX - platforma pro RIA[4] aplikace.

Uvedené informace jsou čerpány z [11] a [16].

Tvorba GUI

Pro tvorbu GUI nabízí Java dvě knihovny. Starší knihovna AWT (Abstract Windowing Toolkit) se již v dnešní době téměř nevyužívá, avšak je stále k dispozici v rámci Java Core API. Novější knihovna JFC Swing (Java Foundation Classes), která se také nachází v Java Core API vychází z AWT. Swing přináší nové druhy komponent a také rozšiřuje stávající možnosti komponent z AWT. [10, 15]

[2] API - Application Programming Interface (aplikační programové rozhraní).
[3] Nejedná se o úplný seznam dílčích platforem.
[4] RIA - Rich Internet Applications. Internetové aplikace, které využívají principu desktopových aplikací.

Použité technologie

Pro implementaci jsem využil platformu Java 6 SE a knihovnu Swing pro tvorbu GUI. Dále jsem kromě základních knihoven pro tvorbu aplikací v Java použil následující externí knihovny:

XStream (verze 1.3.1) - knihovna, která slouží k serializaci objektů do formátu XML. Dále umí také opačnou operaci - deserializaci z XML a získání objektu. Více viz [14].

dom4j (verze 1.6.1) - knihovna pro práci s XML, XPath a XSLT. Více viz [12].

Vývojové prostředí

Jako vývojové prostředí jsem použil NetBeans IDE ve verzi 6.8. Toto prostředí jsem si vybral po dřívější zkušenosti s tvorbou různých druhů aplikací.

NetBeans je multiplatformní vývojový nástroj, jenž je poskytován zdarma. Podporuje vývoj aplikací v různých programovacích jazycích - Java, PHP, C/C++, Python atd. Jedná se o velmi variabilní nástroj, protože využívá zásuvných modulů, tzv. pluginů, čímž lze vytvořit vývojové prostředí na míru každému uživateli. Mezi další výhody tohoto prostředí patří např. automatické doplňování kódu, automatické generování kódu, refaktorizace, GUI návrhář, správa projektu apod. Více informací viz stránky produktu [13].

Při práci jsem využil kromě základních modulů pro tvorbu aplikaci v Java také plugin pro tvorbu UML diagramů. Pro tvorbu GUI jsem využil GUI návrháře, který je součástí NetBeans.

Pojmenovávání

Pro pojmenovávání všech prvků zdrojového kódu jsou použita tato pravidla:

balíky - pojmenování je ve formátu: cz.vutbr.fit.BP.XXX, kde „XXX“ je název balíku s požadovanou funkcionalitou.

třídy - první písmeno každého jednoslovného i víceslovného spojení je velké - využití tzv. PascalCase. Např.: DeskGui, Main.

proměnné - první písmeno každého jednoslovného i víceslovného spojení je malé, ostatní začínající písmena jsou velká - využití tzv. CamelCase. Např.: imageWrite, theta.

metody - stejně jako u proměnných je využit CamelCase. Např.: addCaseToDeskGui, countTheta.

Název aplikace

Název aplikace jsem zvolil BPDemoProg - Bakalářská Práce Demonstrační Program.

4.2 Implementace grafického uživatelského rozhraní

Implementaci grafického uživatelského rozhraní lze rozdělit na tři části - grafické rozhraní, zaznamenávání akcí, které byly s uživatelským rozhraním provedeny a grafickou reprezentaci prvků aplikační logiky.

Grafické rozhraní

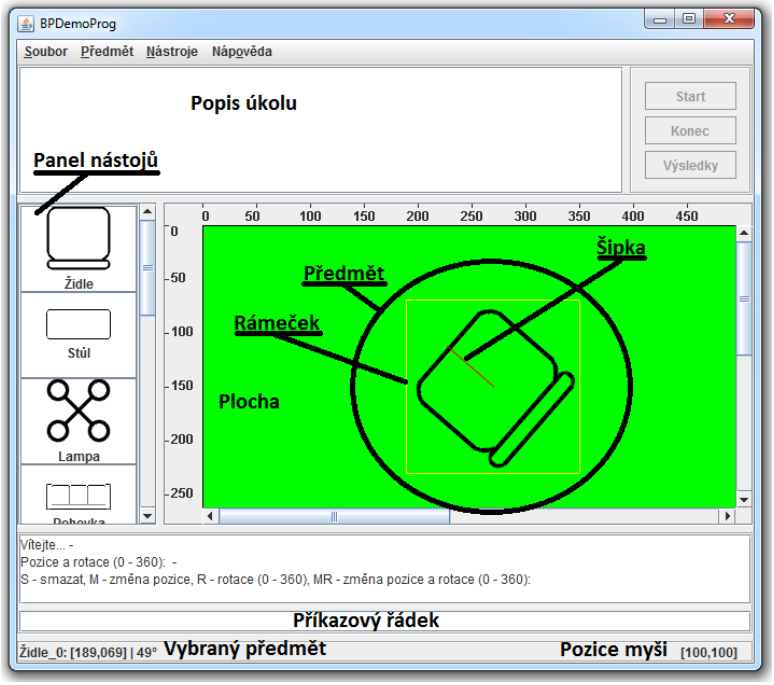
Grafické rozhraní bylo z větší části vytvořeno pomocí GUI návrháře, který je součástí vývojového prostření NetBeans. Jak je výše uvedeno, je využito knihovny Swing.

Grafické rozhraní využívá různé druhy komponent - okna, tlačítka, textové pole atd.

Základní třídou pro tvorbu oken je třída `javax.swing.JFrame`, která dále slouží jako předek pro třídu `RootJFrame` a tato třída je předek třídy `RootJFrameLogging`.

Třída `RootJFrame` je využita jako předek tříd implementující jednotlivá okna, které neposkytují zaznamenávání akcí a pro hlavní okno aplikace. Mezi tyto třídy patří:

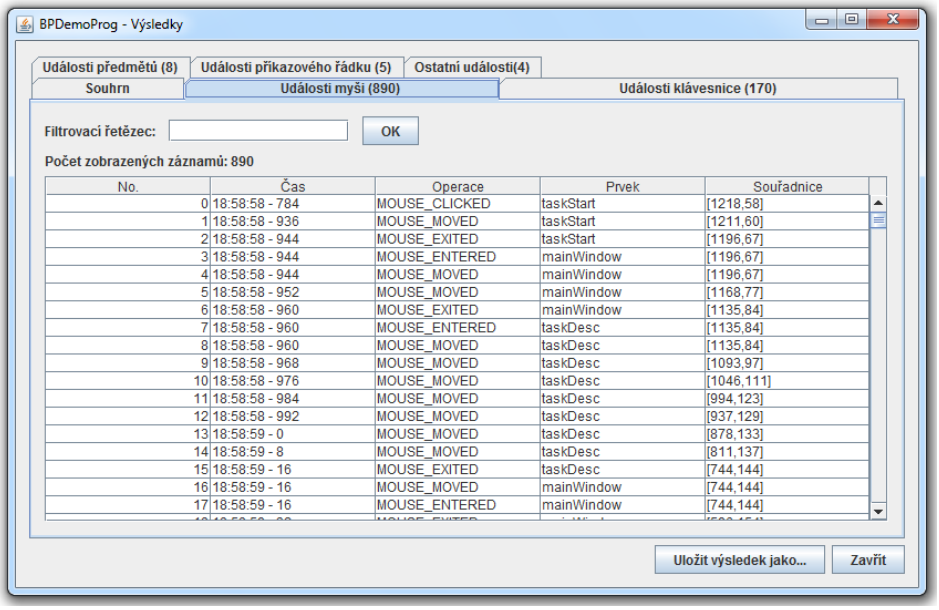
- `MainWindow` - hlavní okno aplikace.
- `ChangeColorsWindow` - okno pro změnu barev.
- `CreateTaskWindow` - okno pro vytvoření popisu úkolu a jeho následného uložení.
- `ResultWindow` - okno pro zobrazení výsledků.



Obrázek 4.1: Hlavní okno aplikace s popisem důležitých prvků.

Třída `RootJFrameLogging` je využita jako předek tříd implementující jednotlivá okna, které poskytují zaznamenávání akcí:

- `AboutWindow` - okno o programu, které obsahuje krátký popis programu.



Obrázek 4.2: Okno pro zobrazení výsledků.

- `AddWindow` - okno pro přidání nového předmětu.
- `HelpWindow` - okno pro zobrazení nápovědy.

Mezi další třídy, které jsou využité pro grafické rozhraní patří:

- `CMDGui` - třída implementující příkazový řádek a historii příkazového řádku.
- `RulerGui` - číselná stupnice - pravítko, pro posuvný panel, na kterém je zobrazena plocha pro přidávání předmětů.
- `ToolBarButton` - třída tlačítka pro zobrazení na panelu nástrojů. Tato třída je předek třídy `AddWindowToolBarButton`, která implementuje tlačítko pro panel nástrojů v okně pro přidání nového předmětu.

Zaznamenávání akcí

Zaznamenávání akcí je povoleno jen v případě plnění úkolu, v jiných případech, jako je tvorba či editace úkolu, nemá smysl. Jak je uvedeno v 3.4, tak je rozlišováno více druhů akcí, a proto je nutné každý druh zaznamenávat samostatně.

Akce myši - třída `MouseLoggingListener` implementuje rozhraní `AWTEventListener`, čímž je schopna přijímat objekty třídy `AWTEvent`, která je kořenovou třídou pro všechny AWT události. Přijatý objekt je následně přetypován na objekt, který je instancí třídy `MouseEvent` a z atributů tohoto objektu je následně vytvořen objekt třídy `MouseLogging`, který reprezentuje jeden záznam události myši, jenž je následně přidán do kolekce záznamů.

Povolení zaznamenávání je provedeno přidáním nového AWTEventListeneru pomocí příkazu `Toolkit.getDefaultToolkit().addAWTEventListener()`, jehož parametry jsou objekt třídy `MouseLoggingListener` a maska příkazů:

`AWTEvent.MOUSE_EVENT_MASK | AWTEvent.MOUSE_MOTION_EVENT_MASK`, která udává, že při události myši (pohyb, stisk tlačítka) se tato událost také předá objektu třídy `MouseLoggingListener`, který vykoná záznam akce.

Pozice myši je odvozena od pozice levého horního rohu vnitřní části hlavního okna aplikace. Hodnotu pozice myši vrací metoda `countRelativeMouse()` třídy `RootJFrame`, která má jeden parametr - objekt třídy `Point`. Hodnota tohoto objektu je získaná metodou `getLocationOnScreen()` třídy `MouseEvent`, která vrátí absolutní pozici myši na obrazovce. Z pozice tohoto bodu je následně přepočtena pozice myši pro uložení do záznamu akce.

Zákaz zaznamenávání je proveden odstraněním daného „posluchače“ pomocí příkazu `Toolkit.getDefaultToolkit().removeAWTEventListener()`, kde parametr je daný objekt.

Záznam je proveden pro každou událost, kterou je `MouseEvent` schopen rozlišit.

Akce kláves - zaznamenávání akcí kláves funguje na stejném principu jako výše zmíněné zaznamenávání akcí myši. Přijatý objekt třídy `AWTEvent` je přetypován na objekt třídy `KeyEvent`, z jeho atributů je vytvořen objekt třídy `KeyLogging` a ten je uložen do kolekce záznamů v dané třídě `KeyLoggingListener`.

Povolení zaznamenávání je provedeno opět přidáním nového AWTEventListeneru, ale s maskou příkazů `AWTEvent.KEY_EVENT_MASK`. Odstranění je opět shodné.

Záznam u kláves je proveden pouze u typu události `KeyEvent.KEY_PRESSED`.

Akce příkazového řádku - objekt třídy `CMDLoggingListener` je prvkem objektu třídy `CMDGui`. Při každém požadavku zadaném přes příkazový řádek je v případě, že je povoleno zaznamenávání, proveden záznam o nastalé akci.

Povolení zaznamenávání se provede příkazem `setEnableLog()` s parametrem `true`.

Zákaz zaznamenávání se provede příkazem `setEnableLog()` s parametrem `false`.

Ostatní akce - do kategorie ostatní patří druhy akcí, které nepatří do žádné z předešlých kategorií - začátek a konec plnění úkolu, okna otevřená během plnění úkolu, změny provedené s nastavením aplikace během plnění úkolu atd.

Zaznamenaná akce je instancí třídy `OtherLogging` a je uchovaná v kolekci záznamů ve třídě `OtherLoggingListener`, která má na starost tento druh akcí.

Povolení zaznamenávání se provede příkazem `setEnableLog()` s parametrem `true`.

Zákaz zaznamenávání se provede příkazem `setEnableLog()` s parametrem `false`.

Protože je důležité znát původce události, která byla následně zaznamenána, je nutné, aby byli veškeří původci událostí nějak vhodně rozlišeni - pomocí jména, proto každý prvek, který může vyvolat záznam akce má přidělené jedinečné jméno. Jméno má každý prvek udělené pomocí metody `setName()`. Jednotlivá jména pro jednotlivé prvky rozhraní jsou uvedena na obrázcích v dodatku C. Předměty na ploše mají jméno udělené ve tvaru: „NÁZEV_ČÍSLO“, kde NÁZEV je název předmětu a ČÍSLO udává kolikátý je to výskyt daného typu během plnění úkolu.

Druhy jednotlivých záznamů jsou uvedeny v dodatku B.

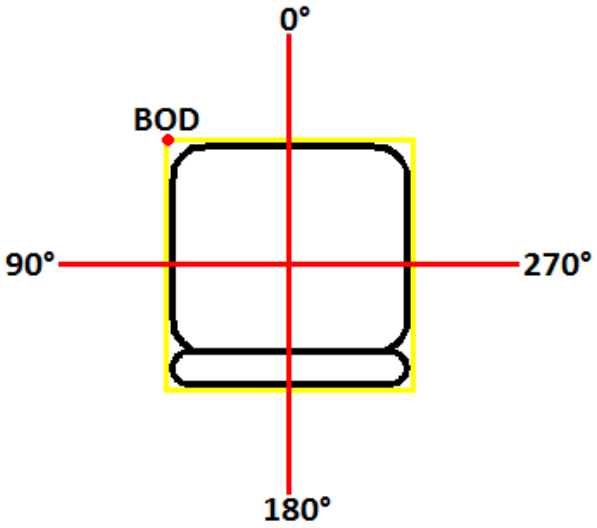
Grafická reprezentace prvků aplikační logiky

Třída `DeskGui` graficky reprezentuje třídu `Desk`, která slouží k uchování prvků třídy `Case`. Třída `Case` je reprezentovaná třídou `CaseGui`.

Třída `DeskGui` implementuje funkcionalitu nutnou ke korektní práci s prvky třídy `CaseGui`.

Třída `CaseGui` implementuje veškeré mechanismy pro práci s předmětem - vykreslování předmětu, změnu parametrů, chování myši při práci s předmětem, mazání předmětu atd. Obrázek 4.3 ilustruje základní vlastnosti předmětu - úhel rotace se nastavuje proti směru hodinových ručiček a bod pro nastavení souřadnic je označen „BOD“. Výše uvedený popis ilustruje obrázek 4.4.

Aby bylo možno s jednotlivými předměty pracovat pomocí všech možných způsobu, tak práce s předmětem třídy `CaseGui` je realizovaná pomocí třídy `CaseWorker`, přes kterou je vedena veškerá agenda spojená s objekty třídy `CaseGui` - přidání, editace a smazání.



Obrázek 4.3: Grafická reprezentace předmětu.

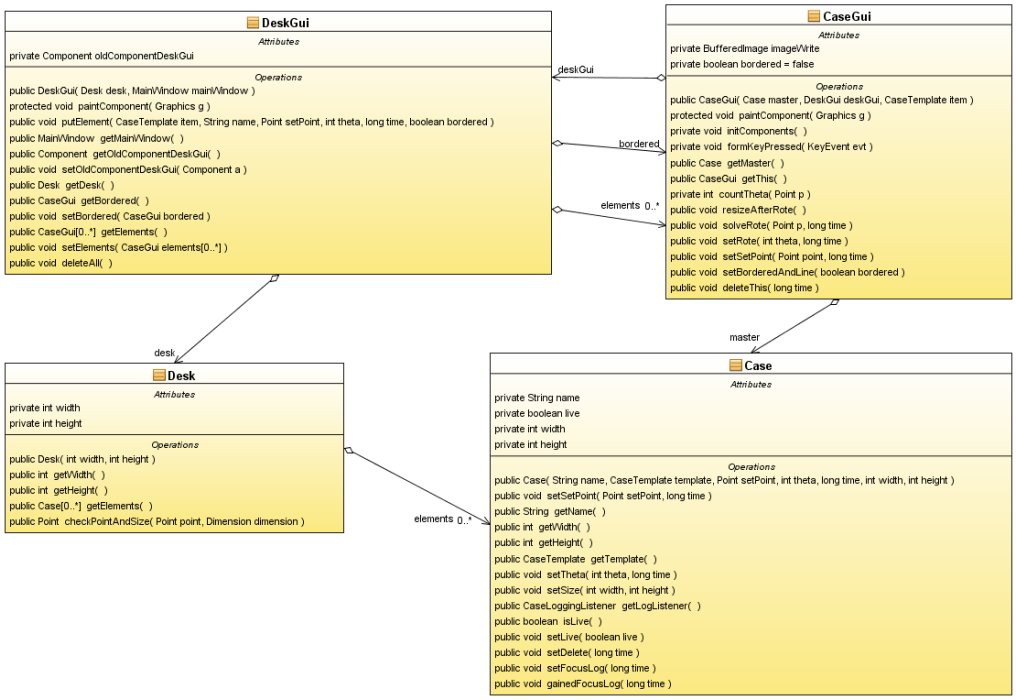
4.3 Implementace aplikační logiky

Implementace aplikační logiky vychází z návrhu uvedeného v části 3.5. Prvním krokem bylo vytvoření třídy `Desk`, která implementuje základní funkcionalitu plochy pro přidávání předmětů. Základní třídou pro reprezentaci předmětu je třída `CaseCore`. Tato třída je využita jako rodičovská třída pro třídu `Case`, která slouží pro práci s předmětem v rámci aplikace. Kolekce prvků třídy `CaseCore` využita k uchování předmětů v úkolu, který je definován třídou `Task`.

Třída `Case` obsahuje prvky třídy `CaseTemplate`, který definuje typ předmětu - jeho rozměry, obrázek, název. Třída`Case` dále obsahuje prvek třídy `CaseLoggingListener`, který provádí zaznamenávání akcí provedených s daným předmětem. Záznam akce implementuje třída `CaseLogging`.

Pro práci s úkolem slouží třída `TaskWorker`, která zastřešuje práci s úkolem - editaci, tvorbu, plnění úkolu.

Výše uvedený popis ilustruje obrázek 4.5.



Obrázek 4.4: Diagram tříd grafické reprezentace prvků aplikační logiky.

Načítání dat

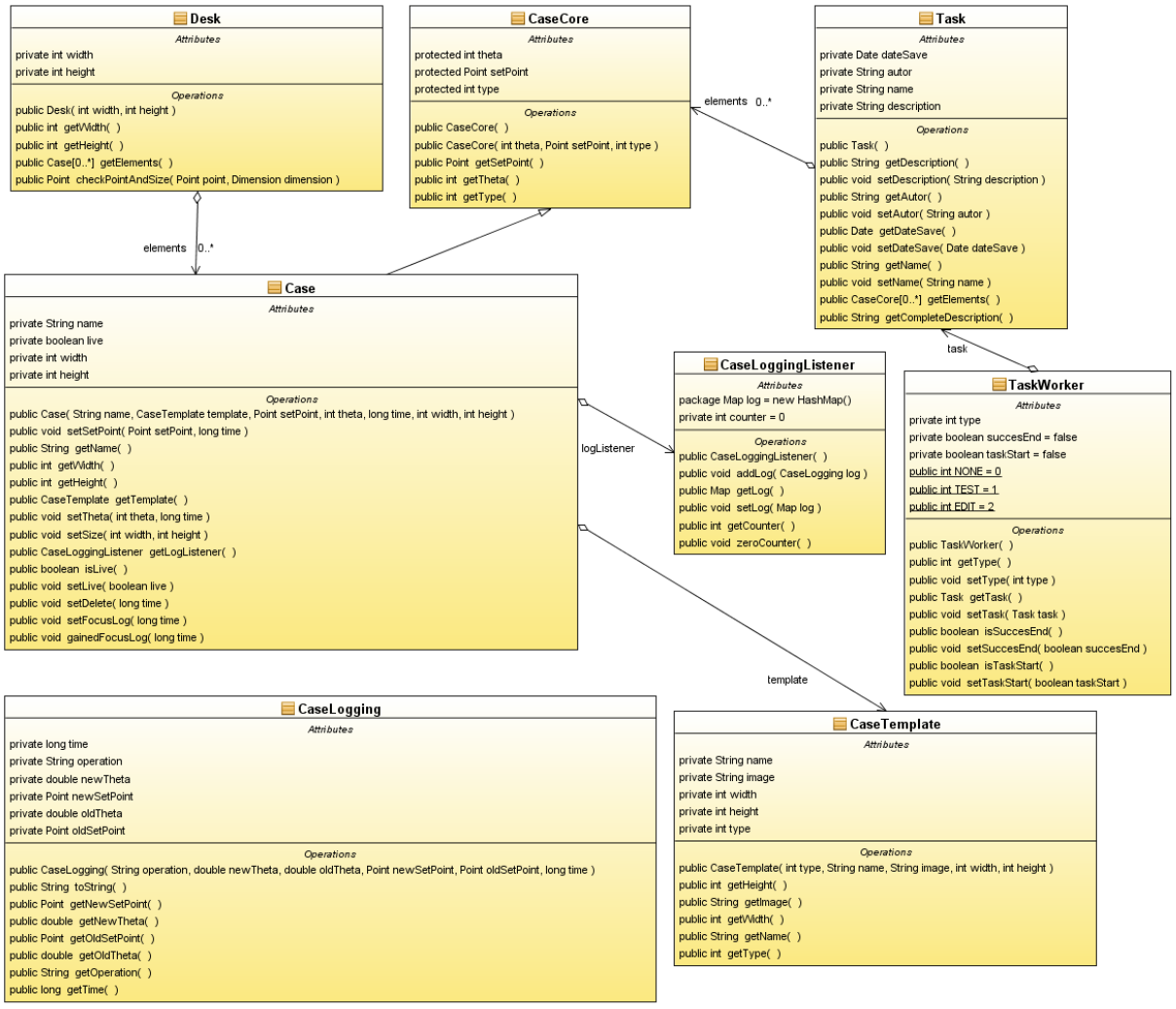
Veškeré konfigurační soubory nutné k práci s aplikací jsou uloženy ve formátu XML, v tomto formátu jsou také uloženy úkoly a výsledky testů. Při ukládání i načítání je využito serializace a deserializace objektů do XML souboru. Pro snažší práci je využita třída `XMLWorker`, která implementuje metody pro načítání a ukládání.

Konfigurační soubor pro nastavení aplikace se jmenuje `settings.xml` a obsahuje veškeré nastavení aplikace - vzhled, barvy, cestu k nápovědě, cestu k adresáři s daty předmětů a cestu ke konfiguračnímu souboru předmětů. V základní verzi jsou veškeré cesty nastaveny na hodnotu `default`, což znamená, že data jsou čtena z jar balíku, který je obsahuje. Pouhou editací těchto cest lze vytvářen nové předměty a jinou nápovědu bez nutnosti zasahování do jar balíku.

Konfigurační soubor `settings.xml` je uložen ve stejném adresáři jako aplikace. Při prvním spuštění je automaticky vytvořen s defaultním nastavením. Pokud během načítání konfiguračního souboru dojde k chybě, tak je tento soubor nahrazen souborem s defaultním nastavením.

Konfigurační soubor s předměty obsahuje název předmětu, rozměry, cestu k obrázku a typ - ID, které je v rámci celého souboru jedinečné. Pokud během načítání konfiguračního souboru s předměty dojde k chybě, tak je konfigurační soubor pro nastavení aplikace (`settings.xml`) nahrazen defaultním nastavením a dojde k načtení dat z jar balíku.

V případě, že během načítání úkolu nebo výsledku testu vznikne chyba, tak aplikace ukončí načítání a objeví se chybové hlášení.



Obrázek 4.5: Diagram tříd aplikační logiky.

4.4 Testování aplikace

Tato část se zabývá testováním výsledné aplikace. V rámci implementace byla aplikace testovaná, aby byla zjištěna výsledná funkčnost aplikace. Bylo provedeno několik druhů testů. Popis testů a jejich výsledky jsou popsány v níže uvedeném textu.

Byl proveden uživatelský test aplikace, který sloužil k zjištění, zda vytvořené uživatelské rozhraní a celková uživatelská přívětivost výsledné aplikace je vhodná pro daný demonstrační příklad. Vybraný uživatel, který nikdy s aplikací nepracoval, ji spustil a začal s ní pracovat. Jelikož aplikace obsahuje klasické ovládací prvky, tak byla reakce uživatele na prostředí rychlá a neměl problém s aplikací pracovat - přidávat prvky na plochu, měnit nastavení aplikace a celkově plně využívat funkčnosti aplikace.

Mezi další druh testu patří test běhu aplikace na různých systémových platformách. I když by u Javy nemělo záležet na systému, protože stačí, aby byla k dispozici Java platforma, tak byl proveden test běhu aplikace na těchto operačních systémech, které měli k dispozici Java platformu: Windows XP, Windows 7, Ubuntu 9.10. Na všech systémech

běžela aplikace bezproblémově.

Při práci s aplikací během plnění úkolu vzrůstá paměťová náročnost aplikace, protože jsou zaznamenávány veškeré uživatelské akce. Tyto záznamy jsou uloženy v operační paměti, ale v dnešní době kapacity operačních pamětí u standardních sestav se nejedná o významný problém.

Na přiloženém CD jsou k dispozici jednoduché ukázkové úkoly, které prezentují, jak může vytvořený úkol vypadat. Dále přiložené CD obsahuje uložené výsledky ukázkových úkolů.

Aplikace je celkově využitelná k záměru, ke kterému byla vytvořena, a to k demonstrování různých druhů uživatelského rozhraní a zaznamenávání uživatelských akcí provedených během plnění úkolu. Možnosti dalšího vylepšení a rozšíření jsou uvedeny v závěrečné kapitole.

Výsledky ukázkových úkolů

Jak je výše uvedeno, tak na přiloženém CD jsou k dispozici ukázkové úkoly a výsledky jejich testování. Tabulka 4.1 uvádí základní informace o výsledcích provedených testů s ukázkovými úkoly. V tabulce je uveden název úkolu, doba trvání vykonávání úkolu a počty jednotlivých záznamů. Pro zobrazení detailnějších informací je nutné otevřít požadovaný soubor s výsledky v prohlížeči výsledků, který je součástí aplikace.

Úkol	**Doba trvání (ms)**	**Myš**	**Klávesnice**	**Příkazový řádek**	**Předměty**	**Ostatní**
Úkol1	45832	2109	59	2	154	4
Úkol2	80933	3621	76	3	474	6
Úkol3	47814	1349	77	9	74	4
Úkol4	37830	2074	46	4	210	3
Úkol5	50677	1014	83	3	25	5

Tabulka 4.1: Výsledky ukázkových úkolů - počty jednotlivých záznamů a doba trvání úkolu.

Kapitola 5

Závěr

Cílem bakalářské práce bylo vytvoření demonstračního programu, který bude demonstrovat použití různých druhů uživatelského rozhraní na osobním počítači při řešení zadaného úkolu, dále pak během řešení daného úkolu zaznamenávat uživatelské akce, které byly provedeny během plnění úkolu. Tento cíl byl splněn.

Poznatky z prvního bodu zadání, které se týkají prostředků pro tvorbu uživatelských rozhraní, jsou uvedeny ve druhé kapitole. Problematiku druhého a třetího bodu zadání popisuje kapitola třetí, která uvádí analýzu problému a návrh demonstračního programu. Zbývající body zadání jsou popsány v předposlední kapitole. Tato kapitola uvádí popis implementačních prostředků, postup implementace jednotlivých částí aplikace a dále uvádí testování výsledné aplikace.

Výsledná aplikace je aplikace s grafickým uživatelským rozhraním, která má jedno hlavní okno a více sekundárních oken. Tato aplikace je vypracovaná v rozsahu zadání a lze ji využít k výukovým účelům nebo jako základ pro další vývoj. V níže uvedeném textu je uvedeno několik možností směru dalšího vývoje.

První možností je tvorba sofistikovanějšího prohlížeče výsledků, který by umožňoval tvorbu grafů, různých druhů statistik, animací či videa ze získaných dat. Druhou možností je vylepšení demonstračního příkladu, který by obsahoval více parametrů, které by uživatel při plnění úkolu musel nastavovat. Například předměty ve 3D a nastavování parametrů v prostoru. Další možností je přidání nových druhů rozhraní, které by byly využitelné pro práci s aplikací. Přidat například ovládání hlasem, gesty těla apod.

Na závěr bych rád uvedl, že jsem se při tvorbě této bakalářské práce dozvěděl nové poznatky z oblasti uživatelských rozhraní a zdokonalil jsem se v programovacím jazyku Java.

Literatura

[1] BS EN ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs). 1998.

[2] ESET NOD32 Antivirus 4 zdarma [online]. `http://www.eset.cz/eset-nod32-antivirus-zdarma`, 2010-05-09 [cit. 2010-05-09].

[3] Ben, S.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[4] Benson, C.; Clark, B.; Nickell, S.: GNOME Human Interface Guidelines 2.2 [online]. `http://library.gnome.org/devel/hig-book/stable/`, 2009-12-03 [cit. 2010-02-04].

[5] Debbie Stone, Caroline Jarrett, Mark Woodroffe, Shailey Minocha: *User Interface Design and Evaluation*. San Mateo, CA, USA: Morgan Kaufmann, 2005, iSBN 0-12-088436-4.

[6] Duce, D.: Portable Network Graphics (PNG) Specification (Second Edition). W3C recommendation, W3C, Listopad 2003 [cit. 2010-05-03], http://www.w3.org/TR/2003/REC-PNG-20031110.

[7] Jenny, P.: *Human-Computer Interaction*. Harlow, UK: Addison Wesley, 1995, iSBN 0-201-62769-8.

[8] Kolektiv autorů: Apple Human Interface Guidelines [online]. `http://developer.apple.com/mac/library/documentation/UserExperience/Conceptual/AppleHIGuidelines/XHIGIntro/XHIGIntro.html`, 2009-08-20 [cit. 2010-02-04].

[9] Kolektiv autorů: Windows User Experience Interaction Guidelines [online]. `http://msdn.microsoft.com/en-us/library/aa511258.aspx`, 2009-12-31 [cit. 2010-02-04].

[10] Kolektiv autorů: The Swing Tutorial [online]. `http://java.sun.com/docs/books/tutorial/uiswing/`, 2010-01-12 [cit. 2010-03-24].

[11] Kolektiv autorů: The Source for Java Developers [online]. `http://www.java.sun.com/`, 2010-03-23 [cit. 2010-03-26].

[12] Kolektiv autorů: dom4j 1.6.1 [online]. `http://www.dom4j.org/dom4j-1.6.1/`, 2010-03-25 [cit. 2010-03-26].

[13] Kolektiv autorů: NetBeans [online]. http://netbeans.org/, 2010-03-25 [cit. 2010-03-26].

[14] Kolektiv autorů: XStream [online]. http://xstream.codehaus.org/, 2010-03-25 [cit. 2010-03-26].

[15] Pavel, H.: *Java grafické uživatelské prostředí a čeština*. České Budějovice, CZ: Kopp, 2006, iSBN 80-7232-237-0.

[16] Pavel, H.: *Učebnice jazyka Java*. České Budějovice, CZ: Kopp, 2006, iSBN 80-7232-115-3.

[17] Pemberton, S.: XHTML^{TM} 1.0 The Extensible HyperText Markup Language (Second Edition). W3C recommendation, W3C, Srpen 2002 [cit. 2010-05-03], http://www.w3.org/TR/2002/REC-xhtml1-20020801.

[18] Quesenbery, W.: Balancing the 5Es: Usability [online]. http://www.wqusability.com/articles/5es-citj0204.pdf, 2004 [cit. 2010-01-25].

[19] Quin, L.: Extensible Markup Language (XML) [online]. http://www.w3.org/XML/, 2010-03-14 [cit. 2010-05-03].

[20] Raymond, E. S.; Landley, R. W.: The Art of Unix Usability [online]. http://www.catb.org/~esr/writings/taouu/html/index.html, 2004-04-18 [cit. 2010-01-26].

[21] Reimer, J.: A History of the GUI [online]. http://arstechnica.com/old/content/2005/05/gui.ars, 2005-05-05 [cit. 2010-01-26].

[22] Sauro, J.: Measuring Usability [online]. http://www.measuringusability.com/, 2010-02-06 [cit. 2010-02-17].

[23] Wikipedia: Punched card [online] – Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/Punched_cards, 2010-01-20 [cit. 2010-01-26].

[24] Wikipedia: Command-line interface [online] – Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/Command-line_interface, 2010-01-21 [cit. 2010-01-26].

[25] Wikipedia: History of computing hardware [online] – Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/Computer_history, 2010-01-25 [cit. 2010-01-26].

[26] Zemčík Pavel: *Tvorba uživatelských rozhraní [studijní opora předmětu Tvorba uživatelských rozhraní]*. Brno, CZ: VUTBr, 2006.

Dodatek A

Obsah CD

Přiložené CD obsahuje:

- Adresář s kompletním NetBeans projektem
- Externí knihovny nutné k běhu aplikace
- Nápovědu k programu
- Programovou dokumentaci - Javadoc
- Přeloženou aplikaci - jar balík
- Technickou zprávu ve formátu pdf
- Testovací data
- Zdrojové soubory technické zprávy pro LaTeX

Dodatek B

Druhy záznamů

Záznamy myši

Záznam akce myši obsahuje následující prvky: čas, textový popis operace, typ operace, název prvku, souřadnice x, souřadnice y. Při zobrazení záznamů není zobrazen typ operace, ale je zobrazen textový popis operace.

Záznam je proveden pro následující operace myši:

Operace	**Popis**
MOUSE_CLICKED	Tlačítko myši je zmáčknuto a uvolněno.
MOUSE_PRESSED	Tlačítko myši je zmáčknuto.
MOUSE_RELEASED	Tlačítko myši je uvolněno.
MOUSE_MOVED	Pohyb myši.
MOUSE_ENTERED	Vstup kurzoru myši do soukromé oblasti komponenty - prvku.
MOUSE_EXITED	Opuštění kurzoru myši do soukromé oblasti komponenty - prvku.
MOUSE_DRAGGED	Změna pozice myši za současného držení tlačítka.
UNKNOWN_TYPE	Neznámá událost myši.

Tabulka B.1: Operace myši.

Příklad záznamů myši (nejsou uvedeny všechny možné druhy operací)[1]:

Čas	**Operace**	**Prvek**	**Souřadnice [x,y]**
22:17:23 - 946	MOUSE_ENTERED	mainWindow	[456,145]
22:17:23 - 946	MOUSE_EXITED	taskDesc	[456,145]
22:17:23 - 946	MOUSE_MOVED	mainWindow	[456,145]
22:17:23 - 962	MOUSE_ENTERED	deskScroll	[448,155]
22:17:23 - 962	MOUSE_EXITED	mainWindow	[448,155]
22:17:23 - 970	MOUSE_MOVED	deskScroll	[445,159]
22:17:23 - 962	MOUSE_MOVED	deskScroll	[448,155]

Tabulka B.2: Příklad záznamů myši.

[1] Čas je ve formátu hodiny:minuty:sekundy - milisekundy.

Záznamy kláves

Záznam akcí kláves obsahuje následující prvky: čas, znak, modifikátory, prvek.

Záznam je proveden jen v případě, že se jedná o typ události `KeyEvent.KEY_PRESSED`.

Příklad záznamů kláves:

Čas	**Prvek**	**Znak**	**Modifikátory**
22:41:39 - 191	taskEnd	'D'	
22:41:41 - 897	caseLampa_0	'S'	
22:41:43 - 76	caseLampa_0	'Delete'	
22:41:44 - 754	cmdLine	'Shift'	Shift
22:41:44 - 769	cmdLine	'Ctrl'	Ctrl+Shift
22:41:44 - 900	cmdLine	'Space'	Ctrl+Shift

Tabulka B.3: Příklad záznamů klávesnice.

Záznamy příkazového řádku

Záznam je reprezentován časem, příkazem a stavem. Stavy jsou rozlišeny dva - `OK`, který indikuje úspěch a `BAD`, který indikuje neúspěch. Příkazy jsou následující:

`cmdBadCommand: PŘÍKAZ` - byl zadán špatný příkaz, status je vždy `OK`.

`addNewCase: PŘÍKAZ` - přidání nového předmětu, status `OK` / `BAD`.

`setSetPoint: PŘÍKAZ` - nastavení bodu, status `OK` / `BAD`.

`setRote: PŘÍKAZ` - nastavení rotace, status `OK` / `BAD`.

`setRoteAndSetSetPoint: PŘÍKAZ` - nastavení rotace a bodu zároveň, status `OK` / `BAD`.

`cmdAdd: PŘÍKAZ` - přidání předmětu pomocí příkazového řádku, status `OK` / `BAD`.

`delete: PŘÍKAZ` - smazání předmětu pomocí příkazového řádku, status `OK` / `BAD`.

PŘÍKAZ - příkaz, který napsal uživatel do příkazového řádku.

Příklad záznamů příkazového řádku:

Čas	**Příkaz**	**Status**
22:57:19 - 584	cmdAdd: add Lustr 1,2 34	BAD
22:57:23 - 39	addNewCase: 1,1	BAD
22:57:24 - 688	addNewCase: 1,1 3	OK
22:57:33 - 729	cmdAdd: add Křeslo 1,1 2	OK
22:57:49 - 128	setSetPoint: M 100, 80 23	BAD
22:57:56 - 937	setSetPoint: M 100,80	OK
22:58:01 - 907	setRoteAndSetSetPoint: MR 1,1 234	OK
22:58:18 - 857	addNewCase: 1,2 34	OK
22:58:24 - 961	cmdBadCommand: chyba	OK

Tabulka B.4: Příklad záznamů příkazového řádku.

Záznamy předmětu

Záznam je reprezentován časem, operace, nový úhel, starý úhel, nový bod, starý bod. Operace jsou následující:

`newCase-NÁZEV` - přidání nového předmětu.

`setSetPoint-NÁZEV` - nastavení bodu.

`setTheta-NÁZEV` - nastavení úhlu rotace.

`deleteCase-NÁZEV` - smazání předmětu.

`setFocusCase-NÁZEV` - nastavení zaměření.

`gainedFocusCase-NÁZEV` - ztráta zaměření.

NÁZEV - název předmětu.

Příklad záznamů předmětů:

Čas	**Operace**	**Nový úhel**	**Starý úhel**	**Nový bod**	**Starý bod**
22:57:33 - 728	newCase-Křeslo_1	2.0	2.0	[1.0,1.0]	[1.0,1.0]
22:57:38 - 364	setFocusCase-Křeslo_1	2.0	2.0	[1.0,1.0]	[1.0,1.0]
22:57:56 - 936	setSetPoint-Křeslo_1	2.0	2.0	[100.0,80.0]	[1.0,1.0]
22:58:01 - 907	setTheta-Křeslo_1	234.0	2.0	[100.0,80.0]	[100.0,80.0]
22:57:38 - 364	gainedFocusCase-Křeslo_0	3.0	3.0	[1.0,1.0]	[1.0,1.0]
22:58:01 - 907	setSetPoint-Křeslo_1	234.0	234.0	[1.0,1.0]	[100.0,80.0]
22:57:24 - 688	newCase-Křeslo_0	3.0	3.0	[1.0,1.0]	[1.0,1.0]
22:58:18 - 856	newCase-Rostlina_0	34.0	34.0	[1.0,2.0]	[1.0,2.0]

Tabulka B.5: Příklad záznamů předmětů.

Ostatní záznamy

Záznam je reprezentován časem, vlastností, novou hodnotou, starou hodnotou. Tento druh záznamu je využít pro ukládání různých vlastností viz tabulka B.7.

Příklad ostatních záznamů:

Čas	**Vlastnost**	**Nová hodnota**	**Stará hodnota**
22:57:10 - 163	task	start	-
22:58:27 - 554	preliminarilyEnd	ask	-
22:58:28 - 438	preliminarilyEnd	OK	-
22:58:28 - 438	task	end	-

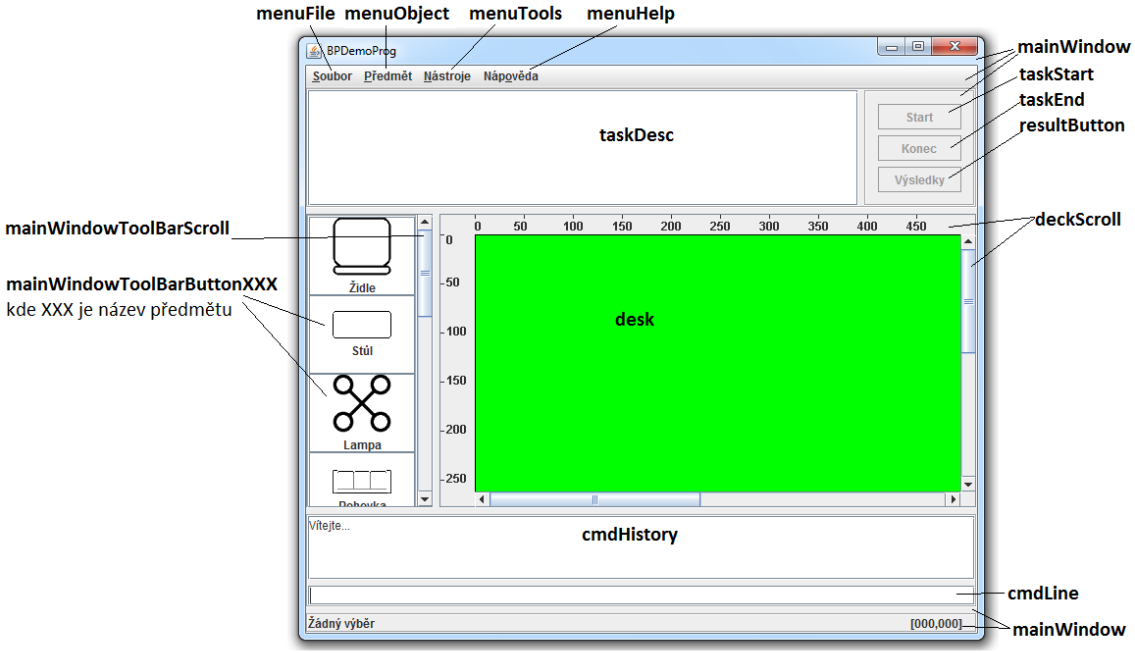
Tabulka B.6: Příklad ostatních záznamů.

Vlastnost	**Nová hodnota**	**Stará hodnota**	**Popis**
addWindow	addButton	-	Stisk tlačítk pro přidání předmětu v okně pro přidání předmětu.
NÁZEV OKNA	close	-	Uzavření okna, jehož jméno udává NÁZEV OKNA.
NÁZEV OKNA	open	-	Otevření okna, jehož jméno udává NÁZEV OKNA.
switchFocusByShortcut	-	-	Stisk klávesové zkratky „Shift + Ctrl + Space“, který je bez změny zaměření předmětu.
switchFocusByShortcut	NEW	OLD	Stisk klávesové zkratky „Shift + Ctrl + Space“, který je se změnou zaměření předmětu. NEW nově zaměřený, OLD původní zaměřený.
alOnTop	NEW	OLD	Nastavení vlastnosti vždy nahoře. NEW - nová hodnota, OLD - stará hodnota.
setDefault	ask	-	Dialogové okno pro nastavení na defaultní hodnoty.
setDefault	ask	OK	Potvrzení nastavení na def. hodnoty.
setDefault	ask	NO	Nepotvrzení nastavení na def. hodnoty.
task	start	-	Start úkolu.
task	end	-	Konec úkolu.
preliminarilyEnd	ask	-	Dialogové okno pro předčas. ukončení úkolu.
preliminarilyEnd	ask	OK	Potvrzení předčas. ukončení.
preliminarilyEnd	ask	NO	Zamítnutí potvrzení předčas. ukončení.
changeProgramStyle	STYL	BAD	Změna stylu aplikace. STYL není podporovaný, tedy není nastaven.
changeProgramStyle	STYL	-	Změna stylu aplikace. STYL je nastaven. Stará hodnota není známa.
changeProgramStyle	NEW	OLD	Změna stylu aplikace. NEW - nová hodnota, OLD - stará hodnota.

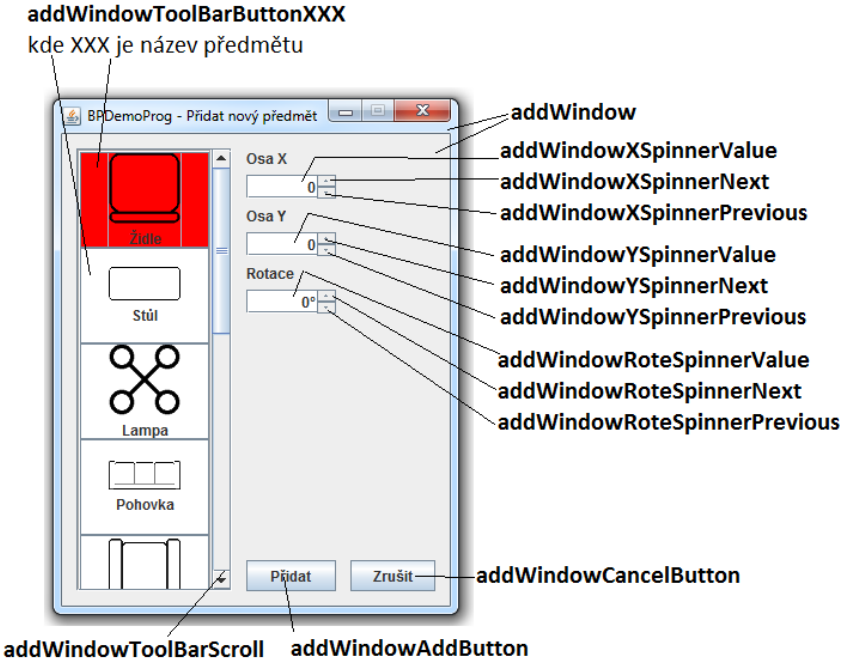
Tabulka B.7: Druhy ostatních záznamů.

Dodatek C

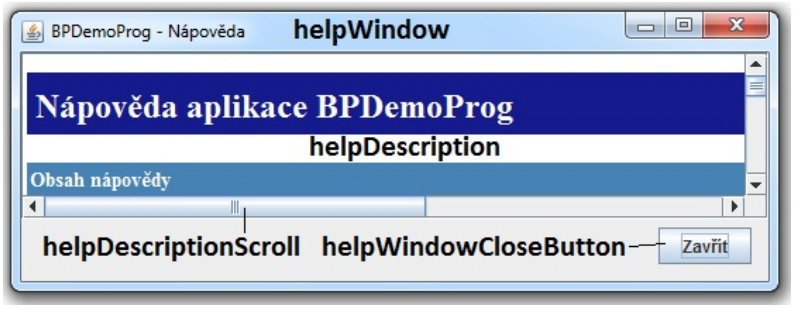
Jména prvků



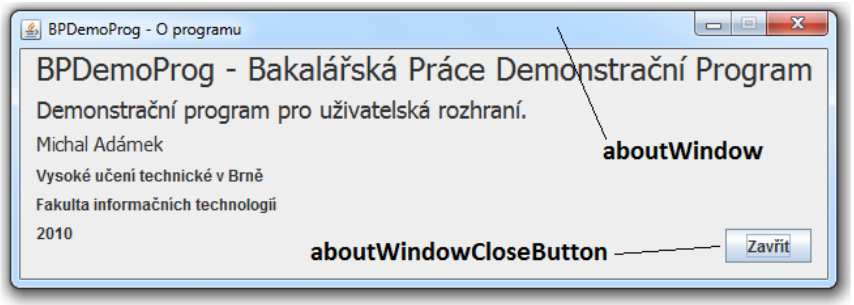
Obrázek C.1: Pojmenování prvků hlavního okna.



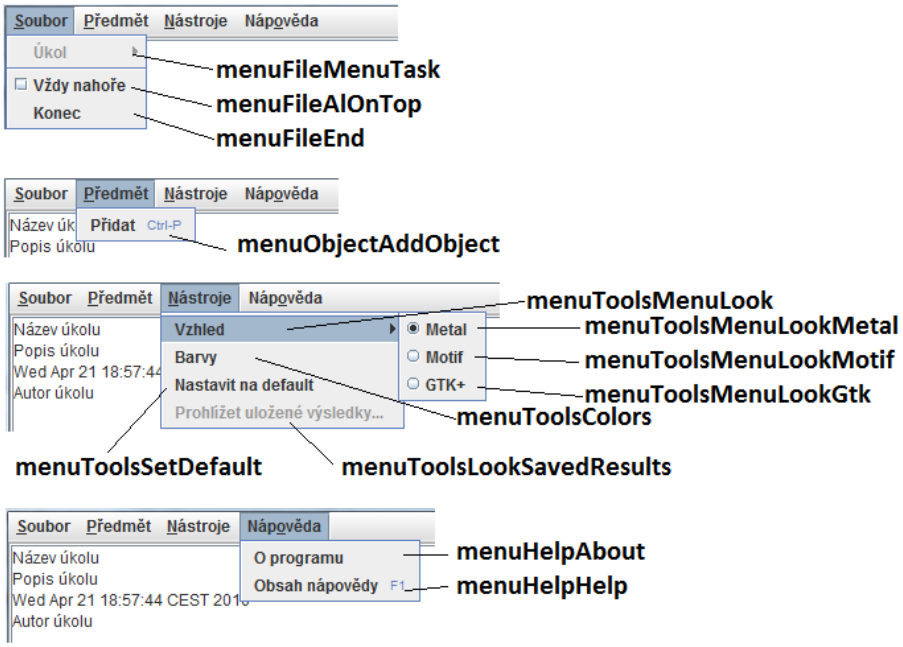
Obrázek C.2: Pojmenování prvků okna pro přidání předmětu.



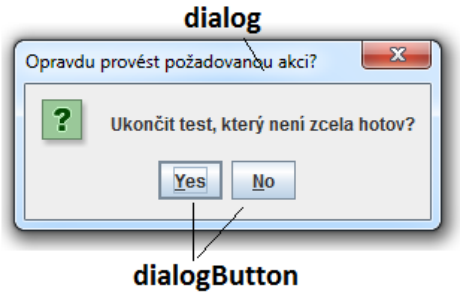
Obrázek C.3: Pojmenování prvků okna pro zobrazení nápovědy.



Obrázek C.4: Pojmenování prvků okna o programu.



Obrázek C.5: Pojmenování prvků všech menu.



Obrázek C.6: Pojmenování prvků dialogu.